Signal Theory and Communications Department

EM based algorithms for Malaria diagnose via crowdsourcing

Carles Díaz Vilor

Supervised by:
Margarita Cabrera Bean
Alba Pagès Zamora

Abstract

We live in a world in which medicine and technology are more united than ever. That is why in the last few years, lots of research groups initially dedicated to the development of technologies, have started to investigate in a field in which progresses are needed in order to protect the humanity against diseases, an this field is the medicine one. This work, following this trend, is focused on one of the diseases that affects the bast majority of tropical countries, Malaria. Along this final Degree Thesis, this disease will be the center of the work, firstly trying to sensitize the reader about its importance and after that, once the objectives have been defined, develop signal processing techniques and algorithms that in the end will count and detect malaria parasites via crowdsourcing, system that is explained in the Introduction chapter.

Resum

Vivim en un món el qual medicina i tecnologia cada cop estan més units. És per això que durant els últims anys, molts grups de recerca que inicialment es dedicaven al desenvolupament de tecnologia, s'han submergit en un camp en el qual es necessiten mes avenços per poder protegir l'ésser humà d'enfermetats, és a dir, el món de la medicina. Aquest treball, seguint aquesta tendència, es centra en una de les enfermetats que més afecta a països de zones tropicals, la Malaria. Durant aquest projecte de final de grau es parlarà sobre aquesta enfermetat i es podrà sensibilitzar el lector de la seva importancia. Un cop els objectius del treball han estat fixats es desenvoluparan tècniques i algorismes de processament del senyal la finalitat dels qual serà la de contar i detectar paràsits de malaria mitjançant crowdsourcing, sistema explicat a la introducció.

Resumen

Vivimos en un mundo en el que medicina y tecnología cada vez van más de la mano. Es por ello que durante los últimos años, muchos grupos de investigación inicialmente dedicados al desarrollo de tecnología, han desembarcado en un campo en el cual se necesitan avances para poder proteger al ser humano de enfermedades, es decir, el campo de la medicina. Este proyecto, siguiendo esta tendencia, se centra en una de las enfermedades que más afecta a países de zonas tropicales, la Malaria. A lo largo de este trabajo final de grado se hablará de esta enfermedad y se podrá sensibilizar al lector de su importancia. Una vez se han fijado los objetivos, se desarrollaran técnicas y algoritmos de procesado de señal cuya finalidad será la de contar y detectar parásitos de malaria mediante crowdsourcing, sistema explicado en la introducción.

Acknowledgments

I want to dedicate this section to the people that has helped me during my final degree thesis. Firstly, I would like to thank my tutors Margarita Cabrera Bean and Alba Pagès Zamora for the knowledge and help that have provided me throughout this time.

Secondly, the majority of the simulations have been done in servers from the ETSETB, so I must thank to the computer support department that helped me with software problems during the project.

Thirdly, I would like to thank the MalariaSpot research group for allowing us to use their images during the work in order to obtain results with real data.

Finally, thanks to my parents for the support that they have given to me during this period.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Malaria is a worldwide infectious disease that affects humans and animals, caused by a parasite belonging to the *Plasmodium* family. After the infection, the parasites, flow through the bloodstream until they arrive to the liver. Once they are in the liver the maturation process yields to the new stage of the parasites, called merozoites. Then these parasites enter the bloodstream and infect the red blood cells, causing its break and infecting more cells. The first symptoms, which are mainly fever, vomiting and headache, usually occur ten days to four weeks after infection, although they my appear even at eight days or up to one year after infection.

It is known that the main transmitter of this disease is the mosquito, but there are other ways, such as from the mother to the fetus or via blood transfusions. This disease is a major health problem in the vast majority of tropical and subtropical countries and The United States Centers for Disease Control and Prevention estimates that 300-500 million cases of malaria occur each year causing the death to more than one million people. Moreover, in some regions of the world, mosquitoes that transmit malaria have developed resistance against insecticides and also the parasites to the antibiotics which have led to the difficulty of controlling both the infection and the spread. Its diagnostic method is by blood examination mostly, demanding trained technicians and consequently a time consuming and expensive task. It is known that this disease can be treated, but the main reason of its mortality is due to that the medicines do not arrive in time or because of the lack of resources in those underdeveloped countries, but maybe the most difficult part of the process is its detection and this work is focused on that, the detection. In here a methodology is presented, that nowadays is widely used for lots of applications, the crowdsourcing.

A crowdsourcing methodology is the one in which a problem is broadcasted through open call asking for contributions from Internet users, providing their solutions, which can be needed services, ideas, decisions etc. The relationship between this methodology and Malaria detection is that by showing a real blood smear, worldwide people can make its own contribution by delivering its opinion (whether if it contains or not parasites) and in the end by taking into account a group of people, make a decision on that sample. Obviously, people who have no idea about how the parasites are, in terms of shape, color and other morphologic properties, do not deliver reliable opinions, and that is why a previous stage of learning is needed. This learning period consists on tagging images whose ground truth (the position of the parasites in the image) is available by the system. In the case that during the learning stage a human achieves good detection results (its decisions are compared with the ground truth in order to see his/her performance) then this human's tags could be used to make decisions in images without ground truth, and in the end have reliable decisions with no experts in Malaria during the process.

The gold standard technique to diagnose this infection is via microscopic examination of Giemsa-stained thick and thin blood films for counting the parasites. However, there are some research groups that use crowdsourcing aiming to detect malaria. The first example that uses crowdsourced data for malaria is The Ozcan Research Group from the UCLA university [12] [13] [14]. This group has its own website and their main objective is to train people in order to detect the malaria infected cells. At the end of each game, the gamer is given a score based on its performance and also a training

feedback in the form of a list of images, containing their false positives and false negatives and in addition, the doubtful cells that they miss-categorized. The aim of this group is to connect large databases to user-friendly games and web-interfaces in order to train and educate medical personal as well as for training machine learning algorithms to automate the diagnosis.

The second example, that will drive this work and indeed motivated it, is the MalariaSpot group [10] [11]. In their project, annotators/gamers are asked to identify malaria parasites in digitized blood smears. This group belongs to the Universidad Politécnica de Madrid (UPM) and has gone one step further in the malaria automated diagnostics by developing human-machine algorithms based on processing crowdsourced data. By means of dedicated on-line platforms, this project offers digitized blood images, via its website, to volunteers who, after a training period, deliver their clicks/tags that are processed by a central decision algorithm. During this work, once the algorithms have been designed, in order to test them with real data, images and the corresponding decisions of the gamers from this group have been used due to a collaboration between research groups from the Universitat Politècnica de Catalunya (UPC) and UPM. The images used in this work that were provided by the MalariaSpot team come from Mozambique, more exactly from the Health Investigation Center of Manhia and were acquired by using a conventional light microscope, a Nokia Xperia Z2 and an on market plastic adapter to attach and align the cellphone camera to the oculars lens of the microscope. The central algorithm has two stages, the first one, corresponding to the clustering, groups the clicks delivered by the gamers, and the second stage makes a decision on each cluster, whether it is a parasite or not. There are different ways of processing data for both stages. For example, in the clustering step, image processing techniques are commonly used. After acquiring and digitizing the image, some filters are used in order to reduce noise and in the end via image segmentation the Red Blood Cells (RBCs), containing the parasites, are delimited. It must be said that segmentation techniques work well with thin blood smears, whereas in this work thick ones will be used. Other methods that try to locate RBCs such as Support Vector Machines (SVM) and Neural Networks (NN). In the other step, the detection one, quorum algorithms are used widespread obtaining acceptable accuracy results, but they may be overtaken by more complex algorithms such as Expectation Maximization (EM) algorithms, which take into account more parameters in order to make a decision on each of the clusters.

As mentioned before, this work will be driven in the same way as the work from the MalariaSpot team. The system will consist of a clustering stage, with the aim of grouping the gamers' tags forming clusters or potential parasites, and then a detection stage, that will provide a binary decision on each cluster representing if that cluster is a parasite or not. For both stages EM algorithms will be developed and tested with both synthetic and real data, looking for a system capable to detect, with high sensitivity and precision rates, malaria parasites. It must be said that both EM algorithms developed for clustering and detection maximize a cost function by executing two steps, the expectation, that calculates the value of a set of variables and the maximization, which calculates the value of a set of parameters in order to maximize the cost function. These two steps, expectation and maximization, will be run until convergence. Due to that the collaboration with the team from Madrid started in 2015, there is some previous work done that has been used along this work, and some theoretical framework tested with synthetic data is published in:

- Margarita Cabrera-Bean, Carles Diaz-Vilor and Josep Vidal, *"Impact of noisy annotators reliability in a crowdsourcing system performance"*, European Signal Processing Conference, August 2016.

This first paper studies how the error probability is degradated when the MAP criteria is applied and the estimates of the reliability parameters of the gamers contain some error. There is a second publication, whose the main difference with respect to the previous one is that in here real data is used and also that the algorithms used are based on EM.

- Margarita Cabrera-Bean, Alba Pagès-Zamora, Carles Diaz-Vilor, Miguel Angel Luengo-Oroz, María Póstigo-Camps, Daniel Cuadrado-Sánchez, *"Counting Malaria Parasites with a two-stage EM based algorithm using crowdsourced data"*, Annual International Conference of the IEEE Engineering in Medicine and Biology Society, July 2017.

One of the most valuable parts of the present work is the application of the algorithms to real data, whose implementation in a system could save time and money when detecting this disease if high accuracy rates are achieved. In addition to doing a system that unifies these two stages, the most innovative part of this work is the on-line implementation of the detection stage.

The project main goals are:

1. To develop clustering techniques in order to identify possible or potential parasites. These clustering techniques are based on EM algorithms and will be tested with either synthetic and real data.
2. To make use of Bayesian techniques to decide if each cluster is a parasite or not. As well as for the clustering, in here EM based algorithms will be developed and tested with both real and synthetic data.
3. To evaluate the quality of the algorithms based on parameters such as the specificity and the precision. The higher these values are, the better the algorithm performs.
4. To develop an on-line implementation of the detection stage in order to emulate a real system that receives data constantly.

Apart from the objectives of the work, another important part is to accomplish some requirements and specifications. As always, a system should achieve small error rates for having good results and be useful, and this is not going to be an exception, so the error probabilities measured during the work should be low. Besides that probabilities, other metrics such as false positives, true positives, false negatives and true negatives are going to be measured, allowing us to calculate the sensitivity and precision of the system, which are two of the most used metrics in medicine. Not only the rates are important, but also the fast performance, which will be determinant in the on-line processing in order to have decisions instantaneously or at least in a short period of time. All the software will be developed in Matlab due to that its easy interface and similarities with other languages that have been studied during my degree at UPC.

The rest of this work is organized as follows. The clustering will be studied in Chapter 2, where different EM-based algorithms to group data are developed. Once these algorithms have been presented, a section of results will show the performance under different scenarios with both real and synthetic data. Chapter 3 is dedicated to the detection stage. As mentioned, in here, EM-based algorithms are developed and tested too, but only with synthetic data. The two stage algorithm is presented in Chapter 4, implementing both stages jointly and showing the performance of the complete system with real data. Chapter 5 is focused on the on-line algorithm, the most innovative part of the work, based on EM and a section of results with both synthetic and real data is included in here too. Finally Chapter 6 contains the budget and Chapter 7 concludes the work.

Chapter 2

Clustering Stage

As it is known, in machine learning there are two procedures whose aim is to study the data, for example group samples in clusters that have similar features and many other applications. The first procedure is the well known Supervised learning, which uses labeled data. The other is the Unsupervised learning, that consists on processing a collection of samples without knowing their category. This section is focused on the second procedure (which includes clustering techniques) because as said in the Introduction chapter, the main purpose of the work is to implement an on-line processing of the data obtained via crowdsourcing in order to make a system capable to detect correctly the parasites given a blood sample. This is possible thanks to a first stage that groups the gamers' tags into different clusters, which are the potential parasites, that will be processed later in the second stage to make a decision in each of them. This section covers all the topics related with the clustering techniques used during this work. These algorithms, in particular two of them, are based on the well known EM, and it has been also tested the clustering via K-Means which is a very common and used method, but it is not going to be explained in here. In order to see which kind of signals have to be processed by the system, the following figure (2.1) shows a real image containing Malaria parasites (left) and the corresponding tags delivered by worldwide gamers with the ground truth (right). In this work, the ground truth will refer to the malaria parasites' position in the images. This chapter is organized as follows. Firstly, the two EM based algorithms are developed. Once the mathematical framework is presented, the algorithms, in addition to K-Means, are tested with both synthetic and real data. The synthetic data is generated artificially and the real one is provided by the MalariaSpot team.

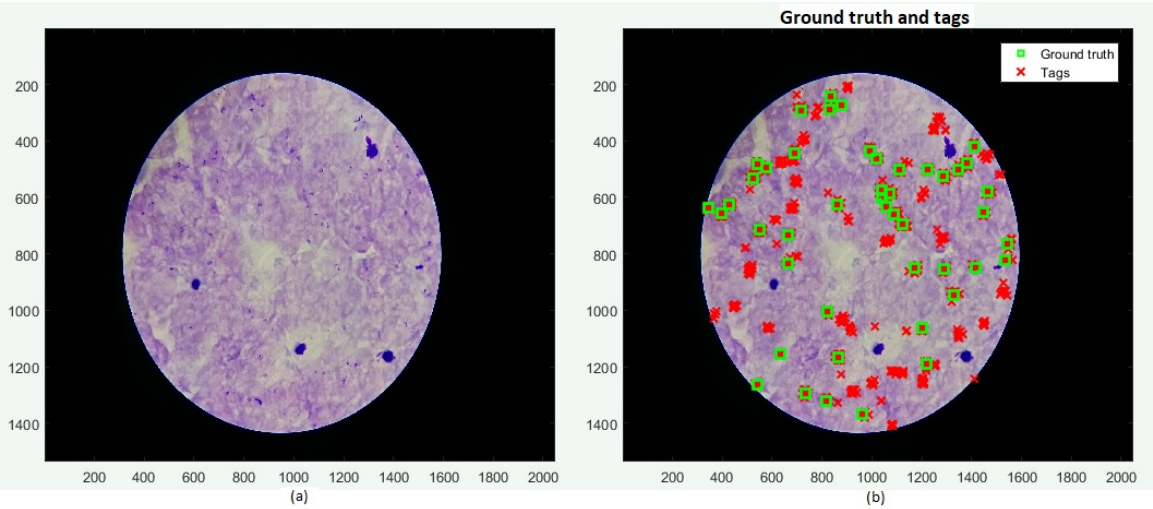


Figure 2.1: Real blood sample (a) and the same sample with 21 worldwide gamers' tags and its ground truth in (b).

2.1 Outliers Expectation Maximization

This section presents the first clustering algorithm, based on the well known EM algorithm. For the rest of the work this algorithm will be referred as Outliers EM (OEM). As said before, the main goal of this stage is to cluster the tags provided by gamers or annotators, that in our real problem these clusters can be considered as potential parasites. The data provided by the annotators is modeled as a mixture of Gaussian components in addition to a random variable (rv) that captures possible outliers. This algorithm will provide soft assignments of data to clusters, estimate the reliability parameters of the gamers and also determine the number of Gaussian components in the model.

Consider a set of R gamers indexed by $r \in \{1, ...R\}$, that provide tags/clicks/instances of a $D \times 1 \in \mathbb{R}^{D\times 1}$ vector, that in this case $D = 2$ due that the work is oriented to images. Each of the instances is modeled by the vector $\boldsymbol{x}_r$, that has the following expression:

$$\boldsymbol{x}_r = a_r \sum_{m=1}^{M} \delta(z_r - m)\boldsymbol{w}_m + (1 - a_r)\boldsymbol{u} \tag{2.1}$$

where $\delta(\cdot)$ denotes the Kronecker delta function, $\boldsymbol{w}_m \sim N(\boldsymbol{\mu}_m\ , \boldsymbol{\Sigma}_m)$ is the m^{th} D-dimensional Gaussian rv and $\boldsymbol{u}$ is a D-dimensional uniformly distributed rv with independent entries and also with known probability density function (pdf) denoted by $g_{\boldsymbol{U}}(\cdot)$ with support $[\mathcal{U}_d^{min}, \mathcal{U}_d^{max}]$ for $d \in \{1...D\}$. Variables $\{a_{r,i}; \forall r, i\} \in \{0, 1\}$ are independent Bernoulli rv with probability $p_r = Pr\{a_{r,i} = 1\}$ for $r = 1...R$. Finally variables $\{z_{r,i}; \forall r\} \in \{1, ..., M\}$ are independent rv with probability $Pr\{z_{r,i} = m\} = \pi_m$. We can asume that all rv variables in (2.1) are independent among them. The purposed model is a mixture of M Gaussians in addition to a uniformly distributed rv with prior probabilities that depends on each gamer. It is easy to see that when $a_r = 1$ the instance provided by annotator r corresponds to one of the M Gaussians. On the other hand, when $a_r = 0$, the instance of the gamer is uniformly distributed and, hence, an outlier. Moreover, p_r is a measure of the annotators' reliability, so when the lower p_r is, the higher the probability that annotator r provides an outlier.

Suppose that each annotator r provides $N_r \in \mathbb{N}$ instances given by $\{\boldsymbol{x}_{r,i} \in \mathbb{R}^{D\times 1}; i = 1, .., N_r\}$, which are independent identically distributed (iid) realizations of $\boldsymbol{x}_r$ in (2.1). Let $\mathcal{X} = \{\boldsymbol{x}_{r,i}; r = 1, ..., R \text{ and } i = 1, ..., N_r\}$ collect the instances of all annotators. For convenience, two more sets are defined: $\mathcal{A} = \{a_{r,i}; \forall r, \forall i\}$ and $\mathcal{Z} = \{z_{r,i}; \forall r, \forall i\}$. Under the previous assumptions, the likelihood function of the observed data is:

$$f(\mathcal{X}; \boldsymbol{\theta}) = \prod_{r=1}^{R} \prod_{i=1}^{N_r} \left(p_r \sum_{m=1}^{M} \pi_m \mathcal{N}(\boldsymbol{x}_{r,i}; \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m) + (1 - p_r) g_{\boldsymbol{U}}(\boldsymbol{x}_{r,i})\right) \tag{2.2}$$

where $\boldsymbol{\theta}$ represents the set of all unknown parameters, which are:

$$\boldsymbol{\theta} = [\boldsymbol{\mu}_1, ..., \boldsymbol{\mu}_M, vec(\boldsymbol{\Sigma}), ..., vec(\boldsymbol{\Sigma}), \pi_1, ..., \pi_M, p_1, ..., p_R, M]$$

In addition to clustering data, the main objective is to locate the centroids of the clusters, its covariance matrices and the probability of occurrence of each clusters. Moreover, the explained algorithm will provide the annotator's reliability which may be used in the detection stage and also the number of Gaussian components in the mixture.

As a closed-form maximization of (2.2) is not possible, we resort to a numerical solution based on the EM algorithm that estimates the parameters of a Gaussian mixture density in addition to a uniformly distribution and the number of Gaussian components. As it will be seen later, this approach that takes into account possible outliers makes the algorithm more robust achieving better rates in terms of sensitivity and precision. The proposed EM algorithm iterates using both observations $\mathcal{X}$ and a posteriori probabilities of the latent variables, in this case the sets $\mathcal{A}$ and

$\mathcal{Z}$. Once the initialization of the parameters is done, taking into account that the initial estimated number of clusters $\hat{M}_0 >> M$, the algorithm alternates between both steps, expectation (E) and maximization (M), iterating till convergence as explained in Algorithm 1.

Algorithm 1 Outliers EM

1: Initialize the unknown parameter vector $\hat{\boldsymbol{\theta}}^0$
2: **Repeat**
3: *E-Step*: given an estimate $\hat{\boldsymbol{\theta}}^k$, compute the conditional expectation:

$$Q(\boldsymbol{\theta}; \hat{\boldsymbol{\theta}}^k) = \mathbb{E}_{\mathcal{A},\mathcal{Z}}\{\log f(\mathcal{X}, \mathcal{A}, \mathcal{Z}; \boldsymbol{\theta}) | \hat{\boldsymbol{\theta}}_k, \mathcal{X}\} \tag{2.3}$$

4: *M-Step*: obtain the estimate for the next iteration as:

$$\hat{\boldsymbol{\theta}}^{k+1} = \arg\max_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}; \hat{\boldsymbol{\theta}}^k)$$

5: **Until** Convergence: $|Q(\boldsymbol{\theta}; \hat{\boldsymbol{\theta}}^k) - Q(\boldsymbol{\theta}; \hat{\boldsymbol{\theta}}^{k+1})| < \epsilon$

The E-step calculates the value of the a posterior probabilities of the latent variables. Then, making use of the Bayes theorem, the final updates of these variables are:

$$\hat{a}_{r,i}^k = \mathbb{E}\{a_{r,i} | \hat{\boldsymbol{\theta}}^k, \mathcal{X}\} = Pr\{a_{r,i} = 1 | \hat{\boldsymbol{\theta}}^k, \mathcal{X}\} = \frac{\hat{p}_r^k \sum_{m=1}^{\hat{M}^k} \hat{\pi}_m^k \mathcal{N}(\boldsymbol{x}_{r,i}; \hat{\boldsymbol{\mu}}_m^k, \hat{\boldsymbol{\Sigma}}_m^k)}{\hat{p}_r^k \sum_{m=1}^{\hat{M}^k} \hat{\pi}_m^k \mathcal{N}(\boldsymbol{x}_{r,i}; \hat{\boldsymbol{\mu}}_m^k, \hat{\boldsymbol{\Sigma}}_m^k) + (1 - \hat{p}_r^k) g_{\boldsymbol{U}}(x_{r,i})} \tag{2.4}$$

$$\hat{z}_{r,i,m}^k = Pr\{z_{r,i} = m | \hat{\boldsymbol{\theta}}^k, \mathcal{X}\} = \frac{\hat{\pi}_m^k \mathcal{N}(\boldsymbol{x}_{r,i}; \hat{\boldsymbol{\mu}}_m^k, \hat{\boldsymbol{\Sigma}}_m^k)}{\sum_{m=1}^{\hat{M}^k} \hat{\pi}_m^k \mathcal{N}(\boldsymbol{x}_{r,i}; \hat{\boldsymbol{\mu}}_m^k, \hat{\boldsymbol{\Sigma}}_m^k)} \tag{2.5}$$

for $r = 1...R$, $m = 1...\hat{M}^k$ and $i = 1...N_r$. Once the values of the latent variables are calculated, in order to compute the parameter estimates, (2.3) must be developed, and after that the partial derivatives with respect to the parameters must be done, which yields to the following expressions:

$$Q(\boldsymbol{\theta}; \hat{\boldsymbol{\theta}}^k) = \sum_{r=1}^{R} \sum_{i=1}^{N_r} \hat{a}_{r,i}^k \sum_{m=1}^{\hat{M}^k} \hat{z}_{r,i,m}^k \log\left(p_r \pi_m \mathcal{N}(\boldsymbol{x}_{r,i} | \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m)\right) + \sum_{r=1}^{R} \sum_{i=1}^{N_r} (1 - \hat{a}_{r,i}^k) \log((1 - p_r) g_{\boldsymbol{U}}(x_{r,i}))$$

$$\frac{\partial Q(\boldsymbol{\theta}; \hat{\boldsymbol{\theta}}^k)}{\partial p_r} = 0 \Longrightarrow \hat{p_r}^{k+1} = \frac{1}{N_r} \sum_{i=1}^{N_r} \hat{a}_{r,i}^k \tag{2.6}$$

$$\frac{\partial Q(\boldsymbol{\theta}; \hat{\boldsymbol{\theta}}^k)}{\partial \boldsymbol{\mu}_m} = 0 \Longrightarrow \hat{\boldsymbol{\mu}}_m^{k+1} = \frac{\sum_{r=1}^{R} \sum_{i=1}^{N_r} \hat{a}_{r,i}^k \hat{z}_{r,i,m}^k \boldsymbol{x}_{r,i}}{\sum_{r=1}^{R} \sum_{i=1}^{N_r} \hat{a}_{r,i}^k \hat{z}_{r,i,m}^k} \tag{2.7}$$

$$\frac{\partial Q(\boldsymbol{\theta}; \hat{\boldsymbol{\theta}}^k)}{\partial \boldsymbol{\Sigma}_m} = 0 \Longrightarrow \hat{\boldsymbol{\Sigma}}_m^{k+1} = \frac{\sum_{r=1}^{R} \sum_{i=1}^{N_r} \hat{a}_{r,i}^k \hat{z}_{r,i,m}^k (\boldsymbol{x}_{r,i} - \hat{\boldsymbol{\mu}}_m^k)(\boldsymbol{x}_{r,i} - \hat{\boldsymbol{\mu}}_m^k)^T}{\sum_{r=1}^{R} \sum_{i=1}^{N_r} \hat{a}_{r,i}^k \hat{z}_{r,i,m}^k} \tag{2.8}$$

To compute the M-step for the other two parameters, π_m and $\hat{M}^k$, some extra conditions are added when deriving:

$$\hat{\pi}_m^{k+1} = \arg\max_{\pi_m} Q(\boldsymbol{\theta}; \hat{\boldsymbol{\theta}}^k) + \log f(\pi_1, ..., \pi_{\hat{M}^k}) \quad s.t. \;\; \pi_m \geq 0; \;\; \sum_{m=1}^{\hat{M}^k} \pi_m = 1 \tag{2.9}$$

where a negative Dirichlet-type prior is assumed: $f(\pi_1, ..., \pi_{\hat{M}^k}) \propto \exp\left\{ -\frac{L}{2} \sum_{m=1}^{\hat{M}^k} \log \pi_m \right\}$ being $L = D(D+3)/2$ the number of parameters per Gaussian component. The negative Dirichlet prior pushes π_m to be equal either to 0 or to 1, and with the constraint, this prior promotes sparsity in the distribution mixture. Hence, the probability of the m^{th} Gaussian component is computed solving the constrained optimization problem and becomes:

$$\hat{\pi}_m^{k+1} = \frac{\max\{0, \left(\sum_{r=1}^{R} \sum_{i=1}^{N_r} \hat{a}_{r,i}^k \hat{z}_{r,i,m}^k\right) - \frac{L}{2}\}}{\sum_{m=1}^{\hat{M}^k} \max\{0, \left(\sum_{r=1}^{R} \sum_{i=1}^{N_r} \hat{a}_{r,i}^k \hat{z}_{r,i,m}^k\right) - \frac{L}{2}\}} \tag{2.10}$$

The main advantage of (2.10) is that clusters with a reduced number of assigned instances will be eventually annihilated, what means $\pi_m = 0$ (because of that, it is recommended to initialize $\hat{M}^0 >> M$), and after that there is an implicit realignment of the non-zero components in the mixture. In the end, the estimated number of Gaussian components will be the ones such that their probability is different to zero. Additionally, at each iteration the algorithm calculates the value of a cost function that penalizes mixtures with a high number of parameters. The function is the following one:

$$\mathcal{L}(\mathcal{X}, \hat{\boldsymbol{\theta}}^k, \hat{M}^k) = -Q(\boldsymbol{\theta}; \hat{\boldsymbol{\theta}}^k) + \frac{L\hat{M}^k}{2} \log\left(\sum_{r=1}^{R} \sum_{i=1}^{N_r} \hat{a}_{r,i}^k\right)$$

Overall, this criterion is used to terminate the EM iterations once convergence is achieved via $\mathcal{L}(\mathcal{X}, \boldsymbol{\theta}^k, \hat{M}^k)$, and it is also used to calculate the final parameter estimates as the ones that minimize the previous function. Then:

$$k_{min} = \arg\min_k \mathcal{L}(\mathcal{X}, \hat{\boldsymbol{\theta}}^k, \hat{M}^k)$$

and the final parameter estimates after the clustering are $\hat{\boldsymbol{\theta}} = \hat{\boldsymbol{\theta}}^{k_{min}}$. Following some literature hints, after $\mathcal{L}(\mathcal{X}, \boldsymbol{\theta}^k, \hat{M}^k)$ converges, the least probable component of the Gaussian mixture is annihilated and the algorithm is run until convergence again. This step is iterated until the number of estimated clusters is equal to one (or to a minimum number of Gaussians). After that, the final parameter estimates are computed as $\hat{\boldsymbol{\theta}} = \hat{\boldsymbol{\theta}}^{k_{min}}$. The following image, 2.2, illustrates the evolution of the previous function with respect to the number of iterations, k in this case. The phenomenon that was explained before can be seen here, when a Gaussian is removed the value of the function increases and then starts decreasing until it reaches the convergence. In this case, the final parameter estimates were the ones at $k = 1057$, when the function has the absolute minimum.

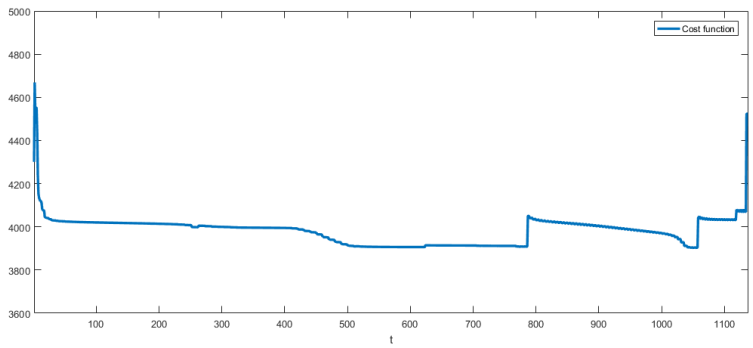


Figure 2.2: Evolution of the function $\mathcal{L}(\mathcal{X}, \boldsymbol{\theta}^k, \hat{M}^k)$ in a single realization.

2.2 Counter-wise Expectation Maximization

This section presents the Counter-wise EM (CEM), which is a particular case of OEM but estimates the same parameters except the reliability of the gamers, which are:

$$\boldsymbol{\theta} = [\boldsymbol{\mu}_1, ..., \boldsymbol{\mu}_M, vec(\boldsymbol{\Sigma}), ..., vec(\boldsymbol{\Sigma}), \pi_1, ..., \pi_M, M]$$

The main difference between both algorithms is that in here outliers are not taken into account, so the final model is the addition of Gaussian components. This implies that there are less variables, such as the ones that define the uniform distribution, the variable $a_{r,i}$ denoting if a tag belongs to a Gaussian or not and finally $p_r = Pr\{a_{r,i} = 1\} = 1$ because $a_{r,i} = 1 \ \forall r, i$. So particularizing the model, it yields to the following equations:

$$\boldsymbol{x}_r = \sum_{m=1}^{M} \delta(z_r - m)\boldsymbol{w}_m \tag{2.11}$$

and the likelihood function ends being:

$$f(\mathcal{X}; \boldsymbol{\theta}) = \prod_{r=1}^{R} \prod_{i=1}^{N_r} \Big(\sum_{m=1}^{M} \pi_m \mathcal{N}(\boldsymbol{x}_{r,i}; \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m) \Big) \tag{2.12}$$

As previously done, because of (2.12) cannot be maximized in a closed form, it is needed to make use of the EM algorithm. The proposed EM algorithm iterates using both observations and latent variables, in this case only the set $\mathcal{Z}$. Once the initialization of the parameters is done, taking into account that the initial estimated number of clusters $\hat{M}_0 >> M$, the algorithm alternates between both steps, expectation (E) and maximization (M) iterating till convergence as in the previous one.

Algorithm 2 Counter-wise EM

1: Initialize the unknown parameter vector $\hat{\boldsymbol{\theta}}^0$
2: **Repeat**
3: *E-Step*: given an estimate $\hat{\boldsymbol{\theta}}^k$, compute the conditional expectation:

$$Q(\boldsymbol{\theta}; \hat{\boldsymbol{\theta}}^k) = \mathbb{E}_{\mathcal{Z}}\{\log f(\mathcal{X}, \mathcal{Z}; \boldsymbol{\theta}) | \hat{\boldsymbol{\theta}}_k, \mathcal{X}\} \tag{2.13}$$

4: *M-Step*: obtain the estimate for the next iteration as:

$$\hat{\boldsymbol{\theta}}^{k+1} = \arg\max_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}; \hat{\boldsymbol{\theta}}^k)$$

5: **Until** Convergence: $|Q(\boldsymbol{\theta}; \hat{\boldsymbol{\theta}}^k) - Q(\boldsymbol{\theta}; \hat{\boldsymbol{\theta}}^{k+1})| < \epsilon$

The E-step calculates the value of the a posteriori probabilities of the hidden variables. Then, making use of the Bayes theorem, the final updates of these variables are:

$$\hat{z}^k_{r,i,m} = Pr\{z_{r,i} = m | \hat{\boldsymbol{\theta}}^k, \mathcal{X}\} = \frac{\hat{\pi}^k_m \mathcal{N}(\boldsymbol{x}_{r,i}; \hat{\boldsymbol{\mu}}^k_m, \hat{\boldsymbol{\Sigma}}^k_m)}{\sum_{m=1}^{\hat{M}^k} \hat{\pi}^k_m \mathcal{N}(\boldsymbol{x}_{r,i}; \hat{\boldsymbol{\mu}}^k_m, \hat{\boldsymbol{\Sigma}}^k_m)} \tag{2.14}$$

Once the a posteriori probabilities have been calculated, in order to compute the parameter estimates, (2.13) must be developed, which yields to the following expression:

$$Q(\boldsymbol{\theta}; \hat{\boldsymbol{\theta}}^k) = \sum_{r=1}^{R} \sum_{i=1}^{N_r} \sum_{m=1}^{\hat{M}^k} \hat{z}^k_{r,i,m} \log \left(\pi_m \mathcal{N}(\boldsymbol{x}_{r,i} | \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m) \right)$$

Similar to the OEM, by computing the partial derivatives, the parameter estimates are:

$$\frac{\partial Q(\boldsymbol{\theta};\hat{\boldsymbol{\theta}}^k)}{\partial \boldsymbol{\mu}_m} = 0 \Longrightarrow \hat{\boldsymbol{\mu}}_m^{k+1} = \frac{\sum_{r=1}^{R}\sum_{i=1}^{N_r} \hat{z}_{r,i,m}^k \boldsymbol{x}_{r,i}}{\sum_{r=1}^{R}\sum_{i=1}^{N_r} \hat{z}_{r,i,m}^k} \tag{2.15}$$

$$\frac{\partial Q(\boldsymbol{\theta};\hat{\boldsymbol{\theta}}^k)}{\partial \boldsymbol{\Sigma}_m} = 0 \Longrightarrow \hat{\boldsymbol{\Sigma}}_m^{k+1} = \frac{\sum_{r=1}^{R}\sum_{i=1}^{N_r} \hat{z}_{r,i,m}^k (\boldsymbol{x}_{r,i} - \hat{\boldsymbol{\mu}}_m^k)(\boldsymbol{x}_{r,i} - \hat{\boldsymbol{\mu}}_m^k)^T}{\sum_{r=1}^{R}\sum_{i=1}^{N_r} \hat{z}_{r,i,m}^k} \tag{2.16}$$

In order to compute the M-step for π_m and $\hat{M}$, the same criterion is used as for OEM. Adding the Dirichlet-type prior to the function in (2.13), we obtain the same as in (2.9) without the term refereed to the uniform distribution. Hence, the probability of the m^{th} Gaussian component is computed solving the constrained optimization problem and becomes:

$$\pi_m^{k+1} = \frac{\max\{0, \left(\sum_{r=1}^{R}\sum_{i=1}^{N_r} \hat{z}_{r,i,m}^k\right) - \frac{L}{2}\}}{\sum_{m=1}^{\hat{M}^k} \max\{0, \left(\sum_{r=1}^{R}\sum_{i=1}^{N_r} \hat{z}_{r,i,m}^k\right) - \frac{L}{2}\}} \tag{2.17}$$

Finally, the way to compute the number of clusters, $\hat{M}$, is similar as in the previous section, but in here the function is different. Taking into account the function that minimizes [9], which is a very cited publication, in order to compute this parameter it is used the Bayesian Information Criterion, that has the following expression:

$$\mathcal{F}(\mathcal{X}, \boldsymbol{\theta}^k, \hat{M}^k) = -\log f(\mathcal{X}|\hat{\boldsymbol{\theta}}^k) + \frac{L}{2}\sum_{m:\pi_m>0}\left(\log\frac{C\pi_m}{12}\right) + \frac{\hat{M}^k}{2}\log\frac{C}{12} + \frac{\hat{M}^k L + M_k}{2}$$

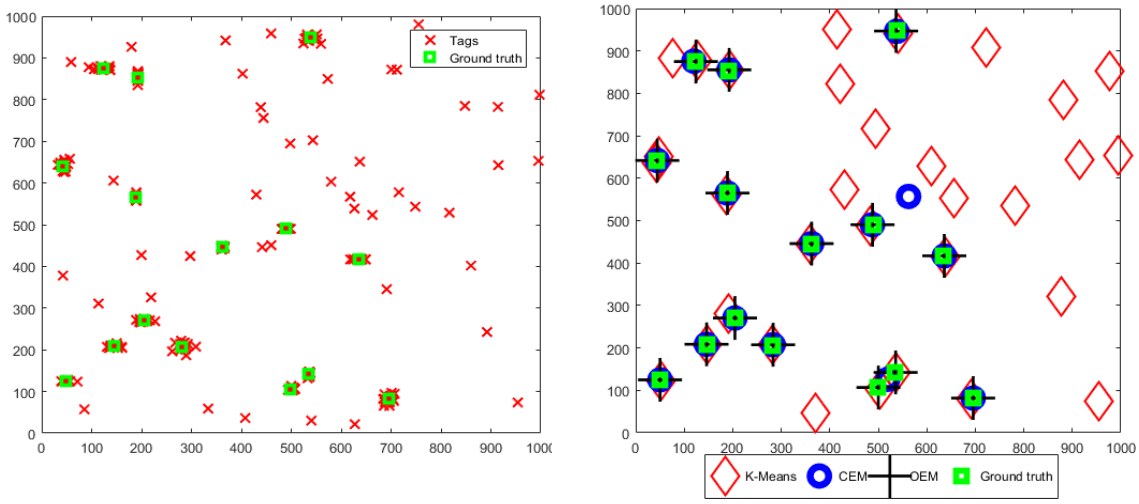
where C is the total number of clicks and $\hat{M}^k$ is the number of Gaussian components in that time instant, k.

2.3 Results

In order to see the performance of the presented algorithms for the Clustering stage, some numerical tests have been made and the well-known clustering algorithm K-Means has been also tested just to compare with the EM-based methods. Two kind of tests have been performed. First of all with synthetic data, and then with real data provided by MalariaSpot. For both simulations, there have been considered a number of gamers of $R = 20$ and $R = 50$ providing two-dimensional instances ($D = 2$).

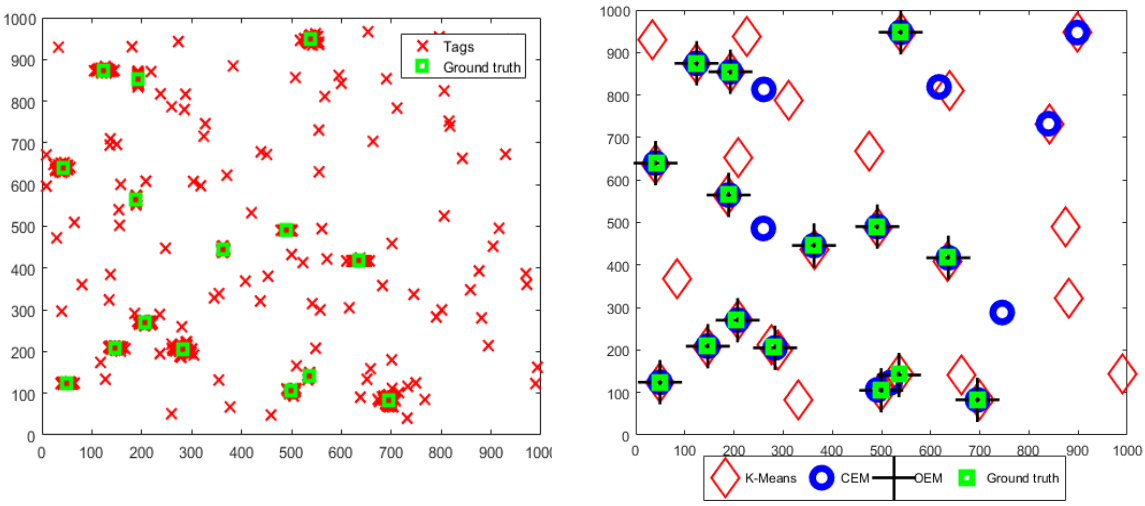
For the synthetic simulations the dimensions of the rectangular (in this case quadratic) are: $\mathcal{U}_1^{min} = \mathcal{U}_2^{min} = 0$ and $\mathcal{U}_1^{max} = \mathcal{U}_2^{max} = 1000$. $N_r \in [13, 17]$ and the number of Gaussian components present in the mixture is $M = 15$ with equal probability. The number of realizations for both synthetic and real data was $N = 1000$.

The first two figures show the tags, ground truth and the cluster centroids found by the three algorithms of one realization with synthetic data. The unique difference between both realizations is that for the first one the number of gamers was $R = 20$ whereas for the second one $R = 50$. The simulation parameters have been selected taken into account that the best algorithm will perform with real data, and so the simulations with non-real data should approximate the real problem.



(a) (b)

Figure 2.3: Image (a) contains the tags given by 20 gamers and the position of the ground truth. Image (b) shows the cluster centroids found by the algorithms using the tags of (a). The data was generated synthetically.



(a) (b)

Figure 2.4: Image (a) contains the tags given by 50 gamers and the position of the ground truth. Image (b) shows the cluster centroids found by the algorithms using the tags of (a). The data was generated synthetically.

It is easy to see that the algorithm that finds more potential parasites is K-Means. This is because the number of clusters is twice the average number of clicks per gamer, and K-Means is an algorithm that does not eliminate components in the mixture, what causes a decrease in the precision. Despite that, if the number of initial centroids, M, is considerably higher than the number of real components, K-Means locates the true parasites. For both realizations, CEM performs well but does not discriminates as well as OEM. For example, CEM has problems when two components are near and there are not many tags, locating one centroid between both, which implies a degeneration of the sensitivity, while OEM does not present this problem. Apart from that, OEM also estimates the number of Gaussian components in the mixture correctly, while CEM presents some errors decreasing the precision, caused by keeping Gaussians that are not in the mixture in reality.

Table 2.1 includes the results in terms of Sensitivity ($S = \frac{TP}{N_p}$ where TP denotes True positives and N_p the number of parasites) and Precision ($P = \frac{TP}{\hat{M}}$ where $\hat{M}$ represents the number of clusters found by the algorithm). As it can be seen, for both configurations (20 and 50 gamers) the algorithm that achieves best performance is the OEM due to that the values of S and P are higher (1 is the perfection). This stage is crucial to achieve a large sensitivity in order not to miss true parasites in the second stage.

Clustering Stage	S 20 gamers	P 20 gamers	S 50 gamers	P 50 gamers
K-Means	0.9789	0.4589	0.9518	0.4759
CEM	0.9502	0.8455	0.9455	0.6211
OEM	0.9975	0.9990	0.9921	0.9995

Table 2.1: Clustering Performance; Sensibility and Precision with synthetic data for 20 and 50 gamers.

Once we have results with synthetic data, the same procedure has been done with real data. For this stage, results of two images are exposed. For each image there will be five figures, the first one showing how is the real image, the second one the same but with the tags and the ground truth for both 20 and 50 gamers and finally the last one will contain the centroids found by OEM and the ground truth too for 20 and 50 gamers.

Image 1:

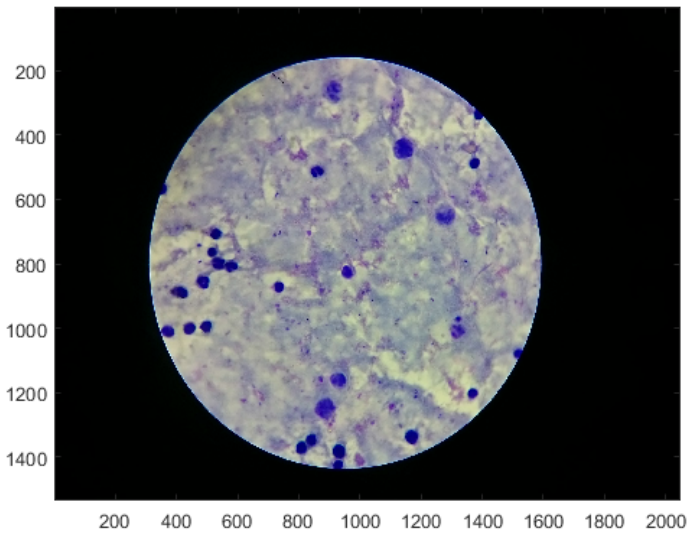
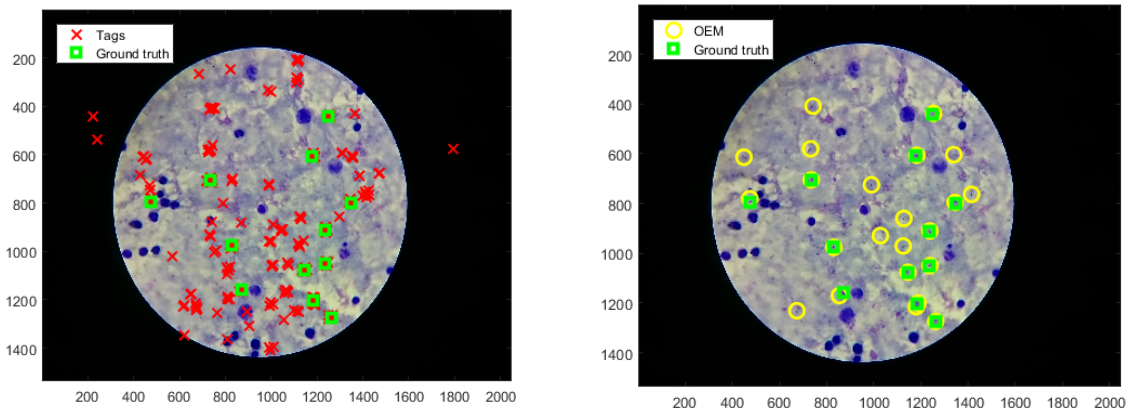
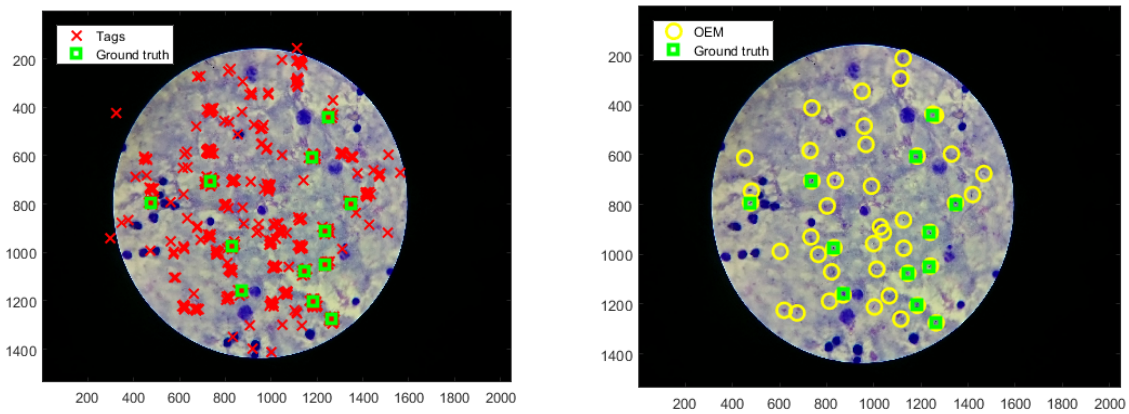


Figure 2.5: Real blood sample containing Malaria parasites



(a) (b)

Figure 2.6: In (a) tags and ground truth of a realization with 20 gamers is shown. In (b) the centroids found by CEM with the ground truth.



(a) (b)

Figure 2.7: In (a) tags and ground truth of a realization with 50 gamers is shown. In (b) the centroids found by CEM with the ground truth.

It is easy to see that for this image the algorithm performs well in terms of S because it finds all the parasites. However, the precision decreases when the number of gamers increases because there are lots of false positives for $R = 50$. This is due to that this image contains lots of artifacts and the gamers prefer to click them, no matter if they are real parasites or not, and so, the higher the number of gamers is, the higher the number of clusters is found. The results in terms of sensitivity and precision for this image are summarized in the following table:

Table 2.2: Clustering Sensitivity&Precision

Clustering Stage	S 20 gamers	P 20 gamers	S 50 gamers	P 50 gamers
K-Means	0.6933	0.2608	0.7025	0.2607
CEM	0.6525	0.3087	0.8108	0.1860
OEM	0.9350	0.4265	0.9992	0.2782

Image 2:

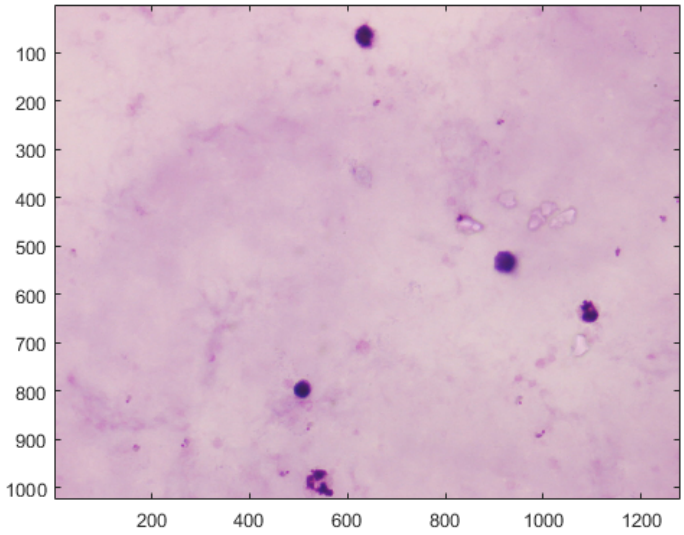
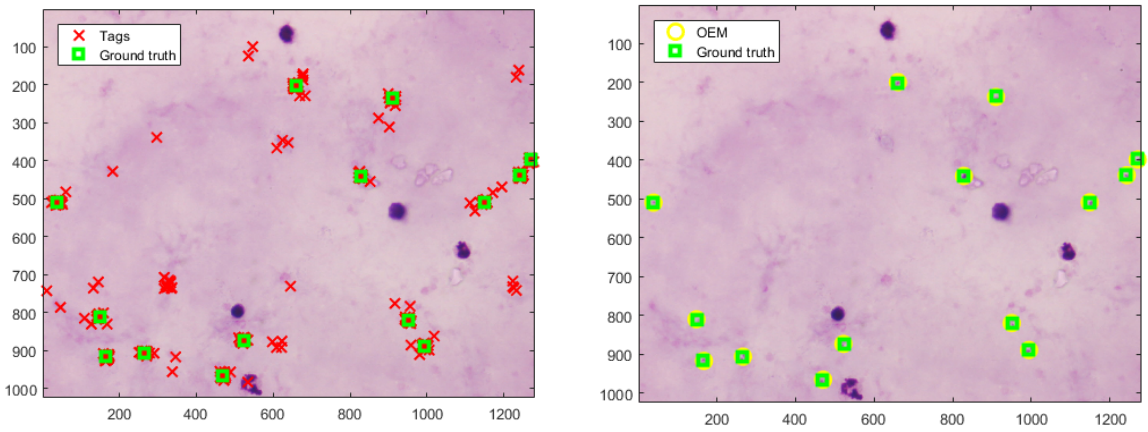
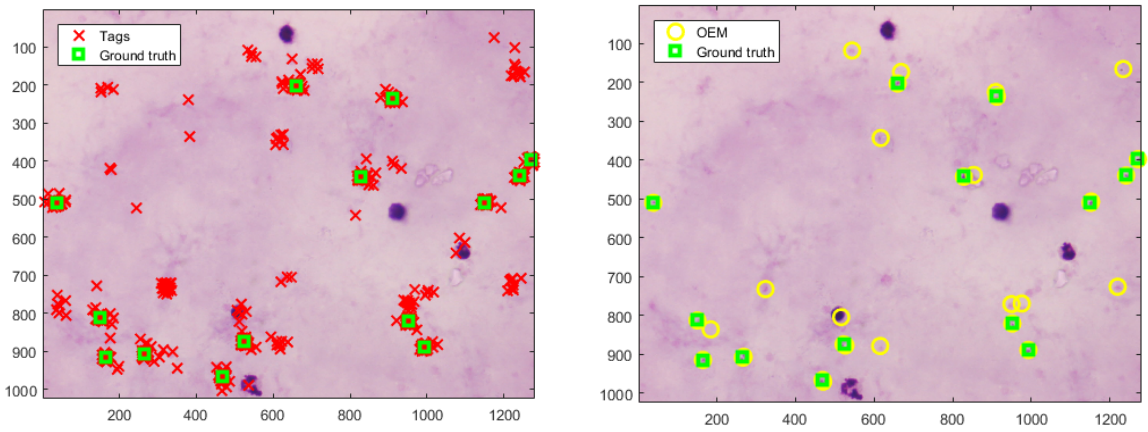


Figure 2.8: Real blood sample containing Malaria parasites



(a) (b)

Figure 2.9: In (a) tags and ground truth of a realization with 20 gamers is shown. In (b) the centroids found by CEM with the ground truth too.



(a) (b)

Figure 2.10: In (a) tags and ground truth of a realization with 50 gamers is shown. In (b) the centroids found by CEM with the ground truth too.

For the second image, we have the same problems as for the first one, but in a lower scale because this image is considered easier to tag due to its color, there are less artifacts and this causes that the gamers' clicks have less variance with respect to the first image. The following table summarizes the performance of the three algorithms for this image:

Table 2.3: Clustering Sensitivity and Precision

Clustering Stage	S 20 gamers	P 20 gamers	S 50 gamers	P 50 gamers
K-Means	0.9900	0.4243	0.9855	0.4211
EMC	0.9391	0.8539	0.9586	0.4935
OEM	0.9989	0.9084	1	0.5150

As it can be see in both tables, 2.2 and 2.3, after 1000 realizations, the best algorithm in terms of sensitivity and precision is the OEM. It allows to detect the parasites (high S rate) and also has the best precision due to the elimination of Gaussian components in the mixture, which causes a decrease in the FP rate, increasing the precision. For the rest of the work, the chosen algorithm for the clustering stage will be OEM. It is important that in this stage the algorithm performs well, in order to achieve high rates in the detection step, which is conditioned for the clustering.

Chapter 3

Detection Stage

The second stage of the system is the Detection step. In here, supposing that several gamers access to the crowdsourcing system and that the clustering has been done, a decision on each of the potential parasites has to be made. There are different ways to compute such decisions, by Majority Voting (MV) for example, but the one that minimizes the error probability is the Maximum a Posteriori (MAP). However, the MAP decision requires the sensitivity and specificity of the annotators, which are unknown parameters for the system and that is why these parameters need to be estimated via EM algorithm. The main problem of this algorithm is that the parameter estimates contain some error with respect to the real ones, so there is an increase in the error probability. A more detailed explanation of the MAP error is given in [4], but in here the work is focused on the EM algorithms. First of all it is going to be developed the EM in a classical way, which in the literature is called Batch EM, and its performance is going to be compared with respect to the MAP. Moreover, another two EM algorithms have been developed taking into account that in the end what we need is a simple algorithm capable to be implemented in an on-line system whose error probability should be acceptable and also having a low computational cost in comparison with the Batch-EM, which is the benchmarking algorithm, that recalculates all the values from the scratch each time instant, without taking advantage from the values calculated previously.

3.1 Batch EM

The first algorithm that has been developed is the classical or batch EM. It estimates the reliability parameters of the gamers (sensitivity and specificity), the prior probabilities and also the unobserved true labels from the gamers' decisions in each cluster, which are the final decisions in each parasite whether it is a positive or a negative. In this case, the observations are given by the vector $\boldsymbol{y}_m \in \mathbb{Z}_2^{R\times 1} m = 1...M$ where R is the number of gamers and M the number of clusters/potential parasites. Each vector $\boldsymbol{y}_m$ is modeled as a mixture of two R-dimensional Bernoulli distributions, the positive with prior probability μ and the negative with prior probability $1-\mu$, associated to the random variable ω, $Pr\{\omega = 1\} = \mu$. Each one of the R gamers has two parameters, $\rho_k^r = Pr\{y_m^r = k|\omega_m = k\}$, for $k \in \{0,1\}$, $r = 1,\ldots,R$. In the end, the algorithm will provide an estimate of the parameter vector: $\boldsymbol{\theta} = [\mu, \boldsymbol{\rho}_1, \boldsymbol{\rho}_0]^T$, where $\boldsymbol{\rho}_k \in \mathbb{R}^{R\times 1}$ for $k = 0, 1$. Then the likelihood function of the observed data, $\mathcal{Y}$, considering M clusters, can be written as:

$$f(\mathcal{Y};\boldsymbol{\theta}) = \prod_{m=1}^{M} \Big(\mu f(\boldsymbol{y}_m|\omega_m = 1;\boldsymbol{\rho}_1) + (1-\mu) f(\boldsymbol{y}_m|\omega_m = 0;\boldsymbol{\rho}_0)\Big) \tag{3.1}$$

The following expressions are the probability density functions of a multivariate Bernoulli distribution, i.e., for $k \in \{0,1\}$ and $m = 1,...,M$. Considering independence between gamers, which is a

common and realistic condition assumed in crowdsourcing, the expressions are :

$$f(\boldsymbol{y}_m|\omega_m = 1; \boldsymbol{\rho}_1) = \prod_{r=1}^{R} (\rho_1^r)^{y_m^r} (1-\rho_1^r)^{(1-y_m^r)}$$
$$f(\boldsymbol{y}_m|\omega_m = 0; \boldsymbol{\rho}_0) = \prod_{r=1}^{R} (\rho_0^r)^{(1-y_m^r)} (1-\rho_0^r)^{y_m^r}$$

Since a closed-form maximization of $f(\mathcal{Y}; \boldsymbol{\theta})$ is not possible, we resort to the EM algorithm. For this, we associate a latent variable $v_m = \{0, 1\}$ to each vector $\boldsymbol{y}_m$ so that if $v_m = 1$ means that the m^{th} cluster is a parasite, and $v_m = 0$ otherwise. The complete likelihood function is given by

$$f(\mathcal{Y}, \mathcal{V}; \boldsymbol{\theta}) = f(\mathcal{Y}|\mathcal{V}; \boldsymbol{\theta}) f(\mathcal{V}; \boldsymbol{\theta}) = \prod_{m=1}^{M} (\mu f(\boldsymbol{y}_m|\omega_m = 1; \boldsymbol{\rho}_1))^{v_m} ((1-\mu) f(\boldsymbol{y}_m|\omega_m = 0; \boldsymbol{\rho}_0))^{(1-v_m)} \quad (3.2)$$

where $\mathcal{V} = \{v_1, ..., v_M\}$ and the parameter vector $\boldsymbol{\theta}$ are estimated by executing two steps at each iteration k upon convergence, as it is summarized in Algorithm 3:

Algorithm 3 EM Algorithm

1: Initialize $\hat{\mu}_0$, $\hat{\boldsymbol{\rho}}_{1,0}$ and $\hat{\boldsymbol{\rho}}_{0,0}$
2: **Repeat**
3: *E-Step*: given an estimate $\hat{\boldsymbol{\theta}}_k$, compute the conditional expectation:

$$Q(\boldsymbol{\theta}; \hat{\boldsymbol{\theta}}_k) = \mathbb{E}_{\mathcal{V}}\{\log f(\mathcal{Y}, \mathcal{V}; \boldsymbol{\theta}) | \hat{\boldsymbol{\theta}}_k, \mathcal{Y}\} = \sum_{\mathcal{V}} f(\mathcal{V}|\mathcal{Y}, \boldsymbol{\theta}) \log f(\mathcal{Y}, \mathcal{V}; \boldsymbol{\theta}) \quad (3.3)$$

4: *M-Step*: obtain the estimate for the next iteration as:

$$\hat{\boldsymbol{\theta}}_{k+1} = \arg\max_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}; \hat{\boldsymbol{\theta}}_k)$$

5: **Until** Convergence: $\sum |\hat{\boldsymbol{\theta}}_k - \hat{\boldsymbol{\theta}}_{k+1}| < \mathcal{E}$

The E-Step calculates the a posteriori conditional probability $\hat{v}_{m,k} = \mathbb{E}\{v_m|\boldsymbol{y}_m; \hat{\boldsymbol{\theta}}_k\} = Pr\{v_m = 1|\boldsymbol{y}_m; \hat{\boldsymbol{\theta}}_k\}$.

$$\hat{v}_{m,k} = \frac{\hat{\mu}_k f(\boldsymbol{y}_m|\omega_m = 1; \hat{\boldsymbol{\rho}}_{1,k})}{\hat{\mu}_k f(\boldsymbol{y}_m|\omega_m = 1; \hat{\boldsymbol{\rho}}_{1,k}) + (1-\hat{\mu}_k) f(\boldsymbol{y}_m|\omega_m = 0; \hat{\boldsymbol{\rho}}_{0,k})} \quad (3.4)$$

Once $\{\hat{v}_{m,k}, \forall m\}$ are calculated, the M-Step maximizes (3.3) where the complete log-likelihood function can be developed as in (3.5) :

$$Q(\boldsymbol{\theta}; \hat{\boldsymbol{\theta}}_k) = \mathbb{E}_{\mathcal{V}}\{log f(\mathcal{Y}, \mathcal{V}; \boldsymbol{\theta})\} = \sum_{\mathcal{V}} f(\mathcal{V}|\mathcal{Y}, \boldsymbol{\theta}) \log f(\mathcal{Y}, \mathcal{V}; \boldsymbol{\theta}) =$$

$$\sum_{m=1}^{M} \hat{v}_m \log \left(\mu \prod_{r=1}^{R} (\rho_1^r)^{y_m^r} (1-\rho_1^r)^{(1-y_m^r)} \right) + (1-\hat{v}_m) \log \left((1-\mu) \prod_{r=1}^{R} (\rho_0^r)^{(1-y_m^r)} (1-\rho_0^r)^{y_m^r} \right) =$$

$$\sum_{m=1}^{M} \hat{v}_m \left(\log \mu + \sum_{r=1}^{R} y_m^r \log \rho_1^r + (1-y_m^r) \log(1-\rho_1^r) \right) +$$

$$+ \sum_{m=1}^{M} (1-\hat{v}_m) \left(\log(1-\mu) + \sum_{r=1}^{R} (1-y_m^r) \log(\rho_0^r) + y_m^r \log(1-\rho_0^r) \right) \quad (3.5)$$

Once the gradient with respect to the parameters has been done, it yields to the following parameter estimates:

$$\frac{\partial Q(\boldsymbol{\theta};\hat{\boldsymbol{\theta}}_k)}{\partial \mu} = 0 \Longrightarrow \hat{\mu}_{k+1} = \frac{1}{M}\sum_{m=1}^{M}\hat{v}_{m,k} \tag{3.6}$$

$$\frac{\partial Q(\boldsymbol{\theta};\hat{\boldsymbol{\theta}}_k)}{\partial \rho_1^r} = 0 \Longrightarrow \hat{\rho}_{1,k+1}^r = \frac{\sum_{m=1}^{M}\hat{v}_{m,k}y_m^r}{\sum_{m=1}^{M}\hat{v}_{m,k}}; \tag{3.7}$$

$$\frac{\partial Q(\boldsymbol{\theta};\hat{\boldsymbol{\theta}}_k)}{\partial \rho_0^r} = 0 \Longrightarrow \hat{\rho}_{0,k+1}^r = \frac{\sum_{m=1}^{M}(1-\hat{v}_{m,k})(1-y_m^r)}{\sum_{m=1}^{M}(1-\hat{v}_{m,k})} \tag{3.8}$$

This algorithm is compared with respect to the MAP, the benchmark method because of its knowledge of all the parameters, and so, the one that achieves the lowest error probability. Moreover, the MV rule has been computed to see the difference between EM and MV too. In order to make these simulations, in this work, the sensitivity and specificity of the gamers are generated following a Beta Distribution, via the mean of the gamers parameters'. The simulation parameters are the following ones: Number of realizations $N = 1000$, $\mathbb{E}\{\boldsymbol{\rho}_1\} = 0.75$, $\mathbb{E}\{\boldsymbol{\rho}_0\} = 0.75$, $\mu = 0.5$, number of gamers $R = 11$, number of clusters $M = 20$. Figure 3.1 shows the Receiver Operating Characteristic (ROC) which represents in one axis the False Negative (FN_r) rate and in the other axis the False Positive (FP_r) rate, corresponding to both error probabilities, $Pr\{e|v_m = 1\}$ and $Pr\{e|v_m = 0\}$ that are calculated following the next expressions:

$$Pr\{e|v_m = 1\} = FN_r = \sum_{\boldsymbol{y}_m \in \mathcal{R}_0}\prod_{r=1}^{R}(\rho_1^r)^{y_m^r}(1-\rho_1^r)^{(1-y_m^r)}$$

$$Pr\{e|v_m = 0\} = FP_r = \sum_{\boldsymbol{y}_m \in \mathcal{R}_1}\prod_{r=1}^{R}(\rho_0^r)^{(1-y_m^r)}(1-\rho_0^r)^{y_m^r}$$

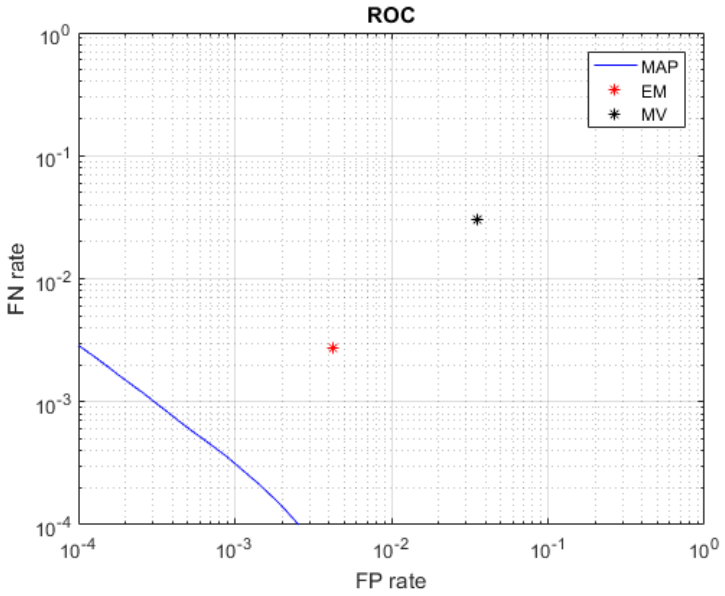


Figure 3.1: FN rate as a function of the FP obtained with synthetic data for MAP, EM and MV as decision systems.

Due to that EM provides an estimate of the parameters, there is an increase in the error probabilities with respect to the MAP, which has te exact value of the parameters and so provides more reliable decisions. Despite that, as our system will not have those values, the decision algorithm that has been chosen for the detection stage is the EM. Apart from achieving good results in terms of error probability, it is a very common and used algorithm when estimating parameters.

3.2 Recursive EM

The second algorithm that has been developed is the Recursive EM. The difference with respect to the Batch EM is that in each time instant, the gamers provide new binary decisions to the same clusters, $\boldsymbol{y}_m(t) \in \mathbb{Z}_2^{R\times 1}$. All vectors $\boldsymbol{y}_m(t)$ are modeled as a mixture of two R-dimensional Bernoulli distributions as well as in the previous algorithm, R representing the number of gamers. Each time instant, gamers give new decisions to the M clusters, so we can define a binary matrix $\boldsymbol{Y}(t) \in \mathbb{Z}_2^{M\times R}$. $\boldsymbol{Y}(t) = [\boldsymbol{y}_1(t), ..., \boldsymbol{y}_M(t)]^T$ where $\boldsymbol{y}_i(t) = [y_i^1(t), ..., y_i^R(t)]^T$. In the end, the algorithm will provide an estimate of the parameter vector: $\boldsymbol{\theta} = [\mu, \boldsymbol{\rho}_1, \boldsymbol{\rho}_0]^T$ previously defined. Then the likelihood function of the observed data, $\mathcal{Y}(t)$ considering M clusters, can be written as:

$$f(\mathcal{Y}(t); \boldsymbol{\theta}) = \prod_{m=1}^{M} \Big(\mu \prod_{l=1}^{t} f(\boldsymbol{y}_m(l)|\omega_m = 1; \boldsymbol{\rho}_1) + (1-\mu) \prod_{l=1}^{t} f(\boldsymbol{y}_m(l)|\omega_m = 0; \boldsymbol{\rho}_0)\Big) \tag{3.9}$$

As in the previous algorithm, (3.9) cannot be maximized in a closed-form, and so that we resort to the EM algorithm. Once the latent variables that allow this maximization are added, the complete likelihood function ends being:

$$\begin{gathered} f(\mathcal{Y}(t), \mathcal{V}; \boldsymbol{\theta}) = f(\mathcal{Y}(t)|\mathcal{V}; \boldsymbol{\theta}) f(\mathcal{V}; \boldsymbol{\theta}) = \\ \prod_{m=1}^{M} (\mu \prod_{l=1}^{t} f(\boldsymbol{y}_m(l)|\omega_m = 1; \boldsymbol{\rho}_1))^{v_m} ((1-\mu) \prod_{l=1}^{t} f(\boldsymbol{y}_m(l)|\omega_m = 0; \boldsymbol{\rho}_0))^{(1-v_m)} \end{gathered} \tag{3.10}$$

As it can be seen in Algorithm 4, the structure is the same as in Algorithm 3 but with just one iteration, whereas for the Batch the algorithm was run until convergence. This makes the Recursive EM a simple algorithm with lower computational cost in comparison with the Batch EM.

Algorithm 4 Recursive EM

1: Initialize $\hat{\mu}$, $\hat{\boldsymbol{\rho}}_1$ and $\hat{\boldsymbol{\rho}}_0$
2: *E-Step*: given an estimate $\hat{\boldsymbol{\theta}}(t)$, compute the conditional expectation:

$$Q(\boldsymbol{\theta}; \hat{\boldsymbol{\theta}}(t)) = \mathbb{E}_{\mathcal{V}}\{\log f(\mathcal{Y}(t), \mathcal{V}; \boldsymbol{\theta}) | \hat{\boldsymbol{\theta}}(t), \mathcal{Y}(t)\} = \sum_{\mathcal{V}} f(\mathcal{V}|\mathcal{Y}(t), \boldsymbol{\theta}) \log f(\mathcal{Y}(t), \mathcal{V}; \boldsymbol{\theta}) \tag{3.11}$$

3: *M-Step*: obtain the estimate for the next iteration as:

$$\hat{\boldsymbol{\theta}}(t+1) = \arg\max_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}; \hat{\boldsymbol{\theta}}(t))$$

The E-Step calculates the a posteriori conditional probability $\hat{v}_m(t) = \mathbb{E}\{v_m(t)|\boldsymbol{y}_m(t); \hat{\boldsymbol{\theta}}(t)\} = Pr\{v_m(t) = 1|\boldsymbol{y}_m(t); \hat{\boldsymbol{\theta}}(t)\}$.

$$\hat{v}_m(t) = \frac{\hat{\mu}(t) \prod_{l=1}^{t} f(\boldsymbol{y}_m(l)|\omega_m = 1; \hat{\boldsymbol{\rho}}_1(t))}{\hat{\mu}(t) \prod_{l=1}^{t} f(\boldsymbol{y}_m(l)|\omega_m = 1; \hat{\boldsymbol{\rho}}_1(t)) + (1-\hat{\mu}(t)) \prod_{l=1}^{t} f(\boldsymbol{y}_m(l)|\omega_m = 0; \hat{\boldsymbol{\rho}}_0(t))} \tag{3.12}$$

Once $\{\hat{v}_m(t), \forall m\}$ are calculated, the M-Step maximizes (3.11). By developing the log likelihood function the following expression is obtained:

$$Q(\boldsymbol{\theta}; \hat{\boldsymbol{\theta}}(t)) = \mathbb{E}_{\mathcal{V}}\{\log f(\mathcal{Y}(t), \mathcal{V}(t); \boldsymbol{\theta})\} = \sum_{\mathcal{V}} f(\mathcal{V}|\mathcal{Y}(t), \boldsymbol{\theta}) \log f(\mathcal{Y}(t), \mathcal{V}; \boldsymbol{\theta}) =$$

$$\sum_{m=1}^{M} \hat{v}_m(t) \log \left(\mu \prod_{l=1}^{t} \prod_{r=1}^{R} (\rho_1^r)^{y_m^r(l)} (1-\rho_1^r)^{(1-y_m^r(l))} \right) + (1-\hat{v}_m(t)) \log \left((1-\mu) \prod_{l=1}^{t} \prod_{r=1}^{R} (\rho_0^r)^{(1-y_m^r(l))} (1-\rho_0^r)^{y_m^r(l)} \right) =$$

$$\sum_{m=1}^{M} \hat{v}_m(t) \Bigg(\log \mu + \sum_{l=1}^{t} \sum_{r=1}^{R} y_m^r(l) \log \rho_1^r + (1-y_m^r(l)) log(1-\rho_1^r) \Bigg) +$$

$$+ \sum_{m=1}^{M} (1-\hat{v}_m(t)) \Bigg(\log(1-\mu) + \sum_{l=1}^{t} \sum_{r=1}^{R} (1-y_m^r(l)) \log(\rho_0^r) + y_m^r(l) \log(1-\rho_0^r) \Bigg) \tag{3.13}$$

Again, computing the derivatives of $Q(\boldsymbol{\theta}; \hat{\boldsymbol{\theta}}(t))$ allows to calculate the final parameter estimates:

$$\frac{\partial Q(\boldsymbol{\theta}; \hat{\boldsymbol{\theta}}(t))}{\partial \mu} = 0 \Longrightarrow \hat{\mu}(t+1) = \frac{1}{M} \sum_{m=1}^{M} \hat{v}_m(t) \tag{3.14}$$

$$\frac{\partial Q(\boldsymbol{\theta}; \hat{\boldsymbol{\theta}}(t))}{\partial \rho_1^r} = 0 \Longrightarrow \hat{\rho}_1^r(t+1) = \frac{\sum_{m=1}^{M} \hat{v}_m(t) \sum_{l=1}^{t} y_m^r(l)}{t \sum_{m=1}^{M} \hat{v}_m(t)}; \tag{3.15}$$

$$\frac{\partial Q(\boldsymbol{\theta}; \hat{\boldsymbol{\theta}}(t))}{\partial \rho_0^r} = 0 \Longrightarrow \hat{\rho}_0^r(t+1) = \frac{\sum_{m=1}^{M} (1-\hat{v}_m(t)) \sum_{l=1}^{t} (1-y_m^r(l))}{t \sum_{m=1}^{M} (1-\hat{v}_m(t))} \tag{3.16}$$

3.3 Incremental Newton

The third algorithm that has been developed is the Incremental Newton, also purposed in [8]. The main difference between the Recursive EM and the Incremental Newton is in the M-Step, where the parameters are estimated by a Newton update. So, the algorithm performs the E-Step as in (3.12) and estimates the parameters as it follows:

$$\hat{\boldsymbol{\theta}}(t+1) = \hat{\boldsymbol{\theta}}(t) + \lambda \boldsymbol{I}(\hat{\boldsymbol{\theta}}(t))^{-1} \mathbb{E}_{\mathcal{V}}\{\nabla_{\hat{\boldsymbol{\theta}}} log f(\mathcal{Y}(t), \mathcal{V}; \boldsymbol{\theta}) | \mathcal{Y}(t)\} \tag{3.17}$$

where λ is a diminishing step size, $\nabla_{\hat{\boldsymbol{\theta}}} log f(\mathcal{Y}(t), \mathcal{V}; \boldsymbol{\theta}) | \mathcal{Y}(t)$ denotes the gradient with respect to $\boldsymbol{\theta}$ evaluated at $\hat{\boldsymbol{\theta}}(t)$ and $\boldsymbol{I}(\hat{\boldsymbol{\theta}}(t))$ represents the Fisher Information Matrix, defined by $\boldsymbol{I}(\boldsymbol{\theta}) = -\mathbb{E}_{\mathcal{V},\mathcal{Y}} \left\{ \frac{\partial^2 log f(\mathcal{Y}(t), \mathcal{V}; \boldsymbol{\theta})}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}^T} \right\}$. Rewriting it in a matrix form:

$$\boldsymbol{I}(\boldsymbol{\theta}) = -\mathbb{E}_{\mathcal{V},\mathcal{Y}} \begin{bmatrix} \frac{\partial^2 log f(\mathcal{Y}(t), \mathcal{V}; \boldsymbol{\theta})}{\partial \mu^2} & \frac{\partial^2 log f(\mathcal{Y}(t), \mathcal{V}; \boldsymbol{\theta})}{\partial \mu \partial \boldsymbol{\rho}_1^T} & \frac{\partial^2 log f(\mathcal{Y}(t), \mathcal{V}; \boldsymbol{\theta})}{\partial \mu \partial \boldsymbol{\rho}_0^T} \\ \frac{\partial^2 log f(\mathcal{Y}(t), \mathcal{V}; \boldsymbol{\theta})}{\partial \boldsymbol{\rho}_1 \partial \mu} & \frac{\partial^2 log f(\mathcal{Y}(t), \mathcal{V}; \boldsymbol{\theta})}{\partial \boldsymbol{\rho}_1 \partial \boldsymbol{\rho}_1^T} & \frac{\partial^2 log f(\mathcal{Y}(t), \mathcal{V}; \boldsymbol{\theta})}{\partial \boldsymbol{\rho}_1 \partial \boldsymbol{\rho}_0^T} \\ \frac{\partial^2 log f(\mathcal{Y}(t), \mathcal{V}; \boldsymbol{\theta})}{\partial \boldsymbol{\rho}_0 \partial \mu} & \frac{\partial^2 log f(\mathcal{Y}(t), \mathcal{V}; \boldsymbol{\theta})}{\partial \boldsymbol{\rho}_0 \partial \boldsymbol{\rho}_1^T} & \frac{\partial^2 log f(\mathcal{Y}(t), \mathcal{V}; \boldsymbol{\theta})}{\partial \boldsymbol{\rho}_0 \partial \boldsymbol{\rho}_0^T} \end{bmatrix}$$

If the log-likelihood has the same expression that (3.13), it is easy to see that the Fisher matrix ends being diagonal because the first partial derivatives with respect to the parameters do not depend on other parameters. All the developments needed to calculate this matrix have been added in Appendix B. The final expression for $\boldsymbol{I}(\hat{\boldsymbol{\theta}})$ is:

$$\boldsymbol{I}(\hat{\boldsymbol{\theta}}) = \begin{bmatrix} \frac{tRM}{\hat{\mu}(1-\hat{\mu})} & 0 & \dots & \dots & \dots & \dots & 0 \\ 0 & \frac{t\mu M}{\hat{\rho}_1^1(1-\hat{\rho}_1^1)} & \dots & \dots & \dots & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & \dots & \frac{t\mu M}{\hat{\rho}_1^1(1-\hat{\rho}_1^R)} & \dots & \dots & 0 \\ 0 & \dots & \dots & \dots & \frac{t(1-\mu)M}{\hat{\rho}_0^1(1-\hat{\rho}_0^1)} & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & \dots & \dots & \dots & 0 & \frac{t(1-\mu)M}{\hat{\rho}_0^R(1-\hat{\rho}_0^R)} \end{bmatrix}$$

Once $\mathbb{E}_{\mathcal{V}}\{\nabla_{\hat{\boldsymbol{\theta}}} log f(\mathcal{Y}(t), \mathcal{V}; \boldsymbol{\theta}) | \mathcal{Y}(t)\}$ has been calculated, whose expressions are in Appendix B, the final estimates of the parameters are:

$$\hat{\mu}(t+1) = \hat{\mu}(t) + \lambda \frac{1}{M} \sum_{m=1}^{M} \left(\hat{v}_m(t) - \hat{\mu}(t) \right) \tag{3.18}$$

$$\hat{\rho}_1^r(t+1) = \hat{\rho}_1^r(t) + \lambda \frac{1}{t\mu M} \sum_{m=1}^{M} \hat{v}_m(t) \sum_{l=1}^{t} \left(y_m^r(l) - \hat{\rho}_1^r(t) \right) \tag{3.19}$$

$$\hat{\rho}_0^r(t+1) = \hat{\rho}_0^r(t) + \lambda \frac{1}{t(1-\hat{\mu}(t))M} \sum_{m=1}^{M} (1 - \hat{v}_m(t)) \sum_{l=1}^{t} \left(1 - y_m^r(l) - \hat{\rho}_0^r(t) \right) \tag{3.20}$$

3.4 Results

In order to evaluate the performance of the algorithms some numerical tests, with synthetic data, have been done. As said before, they are compared with respect to the Batch EM in different scenarios, and in all of them the number of realizations is $N = 1000$. To analyze the results, true/false positives/negatives (denoted by TP, TN, FP and FN) after the detection have been counted in order to compute sensitivity (S) and precision (P). Moreover, the mean squared error (MSE) of the parameters has been calculated too. The plotted errors are:

1. MSE of $\boldsymbol{\rho}_1 = \frac{1}{N} \sum_{n=1}^{N} \frac{1}{R} \sum_{r=1}^{R} (\rho_{1,n}^r - \hat{\rho}_{1,n}^r)^2$
2. MSE of $\boldsymbol{\rho}_0 = \frac{1}{N} \sum_{n=1}^{N} \frac{1}{R} \sum_{r=1}^{R} (\rho_{0,n}^r - \hat{\rho}_{0,n}^r)^2$
3. MSE of $\mu = \frac{1}{N} \sum_{n=1}^{N} (\mu - \hat{\mu})^2$
4. $S = \frac{TP}{Np}$ where Np denotes the number of true parasites.
5. $P = \frac{TP}{M}$ where M represents the number of clusters.

- First scenario: in here, the number of gamers is $R = 15$, the number of clusters is $M = 10$, $\mathbb{E}\{\boldsymbol{\rho}_1\} = 0.8$, $\mathbb{E}\{\boldsymbol{\rho}_0\} = 0.6$ and $\mu = 0.5$, representing equiprobability in each one of the clusters, assumption made for all the scenarios.

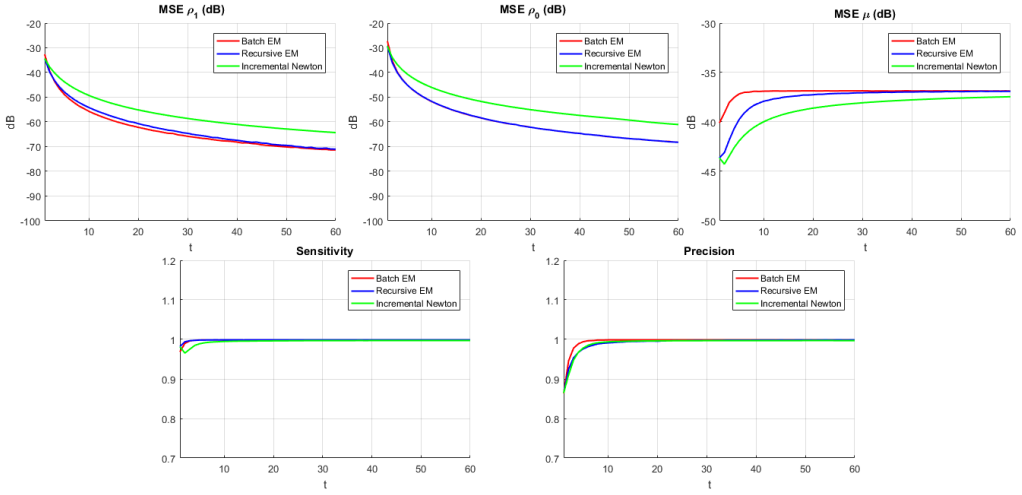


Figure 3.2: The three sub-figures on top, show the evolution of the MSE of the parameters measured un dB over time increases. Under them, sensitivity and precision are plotted.

As it shows the previous figure, both Batch and Recursive EM converge to the same values of MSE approximately when the system has received enough data, for example at $t = 30$. However, the Incremental Newton does not achieve these values because has a slower speed convergence. Looking at the sensitivity and precision, this phenomenon does not appear and the three algorithms provide good rates at early times with little differences.

- Second scenario: for the second simulation, the number of gamers is $R = 7$, the number of clusters $M = 15$ and $\mathbb{E}\{\boldsymbol{\rho}_1\} = 0.9$, $\mathbb{E}\{\boldsymbol{\rho}_0\} = 0.5$.

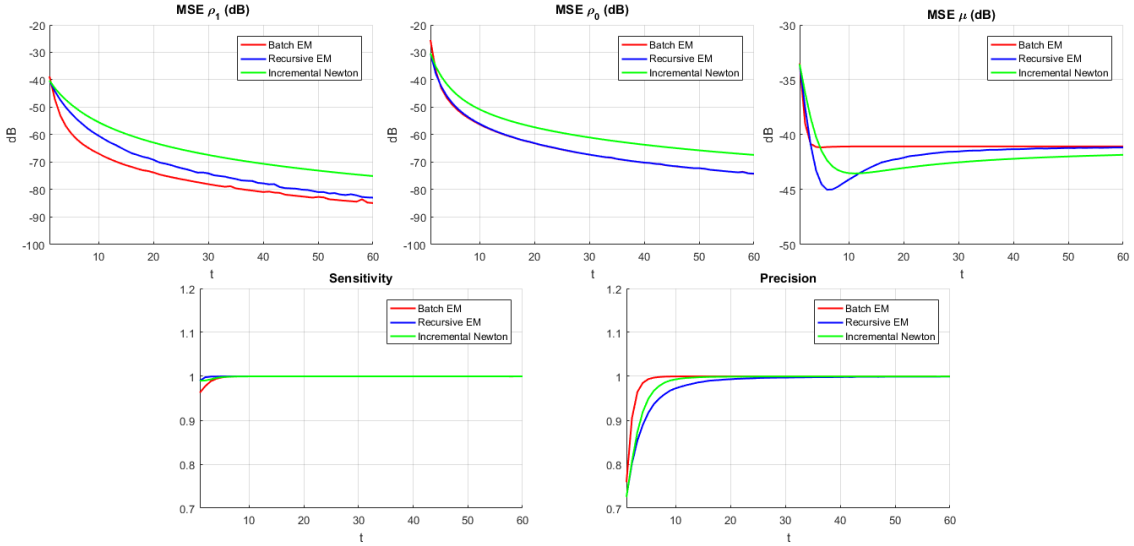


Figure 3.3: The first three sub-figures show the evolution of the MSE of the parameters measured un dB during time. Then the sensitivity and precision are plotted.

The main difference with respect to the first scenario, is that when the number of gamers is lower, the system has less parameters to estimate and so the MSE is lower. Despite that, as the number of clusters has increased but there are few gamers, in comparison with the previous simulation, the precision rate takes more time to converge. However, the sensitivity has a similar behavior because the increase of $\mathbb{E}\{\boldsymbol{\rho}_1\}$ compensates the increase of N_p, and the gamers, despite there are more parasites, provide reliable decisions conditioned to one.

- Third scenario: the parameters of the last simulation are the following ones. The number of gamers $R = 11$, number of clusters $M = 11$, $\mathbb{E}\{\boldsymbol{\rho}_1\} = 0.7$, $\mathbb{E}\{\boldsymbol{\rho}_0\} = 0.7$.

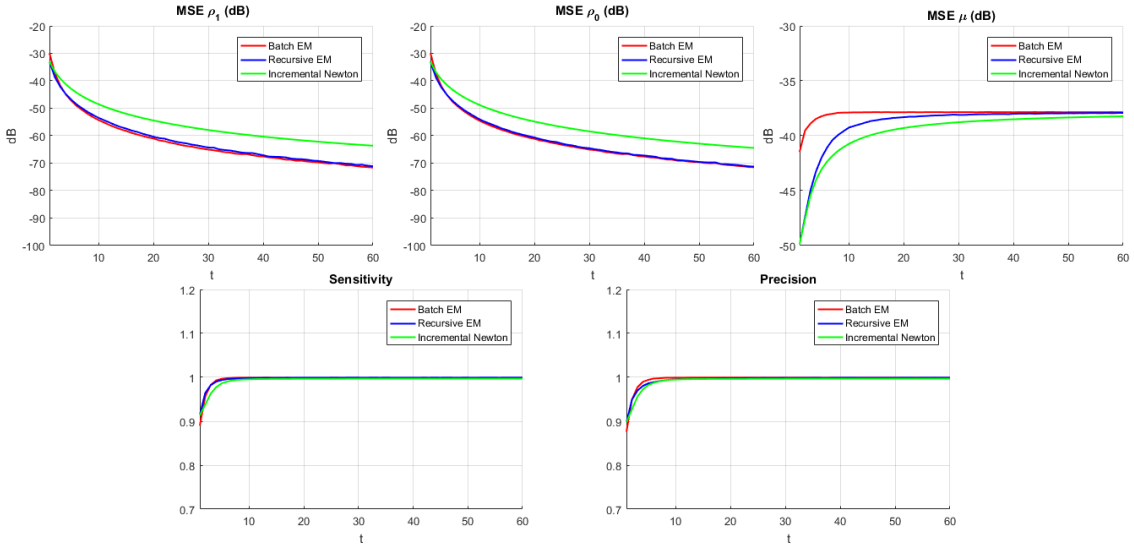


Figure 3.4: The first three sub-figures show the evolution of the MSE of the parameters measured un dB during time. Then the sensitivity and precision are plotted.

Finally, the last scenario is a completely symmetric one. This simulation allows to discard the Incremental Newton for this stage, because it confirms that always has the slowest speed convergence. As it can be seen in the figures, the best situation in terms of MSE happens when there are more clusters than gamers. This is just because the system receives a lot of data and the number of parameters to be estimated is lower than when it occurs the opposite, when the system has lots of gamers but not enough data, what causes an increase of the error.

If we focus on the sensitivity and precision, it is easy to see that all of them have converged approximately at $t = 5$, but before that time instant the best case is when the system has more gamers than clusters. Despite the fact that there is more error in the estimates, due to that the number of gamers is high, it helps to make more correct decisions in each one of the clusters. This is an important conclusion, because when testing the complete system (clustering and detection) with real data, it is better to have more gamers than potential parasites/clusters.

Chapter 4

Two-stage algorithm

As it has been explained in previous sections, the aim of the work is to implement both steps, clustering and detection, in the same system and in an on-line way, but first this combination has been developed off-line. This chapter has unified both stages to make the two step algorithm and it has been tested with real data. First of all R gamers deliver their tags, generating the set of all the clicks $\mathcal{X}$, and then they are processed in the clustering stage via OEM which will provide the position of the centroids or potential parasites. Then, using the Classical/Batch EM the system estimates the unknown parameters and also makes a decision on each of the centroids.

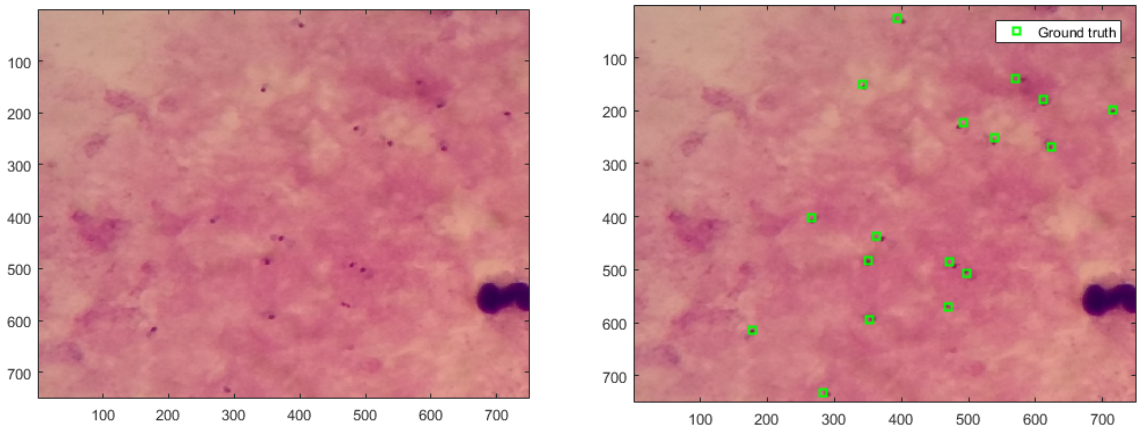
Between both stages, there is an intermediate step. After clustering, low reliable tags, for example the ones whose probability is below a value, are discarded from the set. This means that $\{\boldsymbol{x}_{r,i}; \forall r, i\}$ such that $\hat{a}_{r,i} < 0.5$ are not taken into account, where $\hat{a}_{r,i}$ are calculated as in (2.4) . Then, the decision vectors on each cluster $\boldsymbol{y}_m$ are generated in the following way: $y_m^r = 1$ if there exists a tag $\boldsymbol{x}_{r,i} \in \mathcal{X}$ satisfying:

$$\hat{\pi}_m \mathcal{N}(\boldsymbol{x}_{r,i}; \hat{\boldsymbol{\mu}}_m, \hat{\boldsymbol{\Sigma}}_m) > \hat{\pi}_n \mathcal{N}(\boldsymbol{x}_{r,i}; \hat{\boldsymbol{\mu}}_n, \hat{\boldsymbol{\Sigma}}_n); \forall n \neq m \qquad n, m = 1...M$$
$$\hat{\pi}_m \mathcal{N}(\boldsymbol{x}_{r,i}; \hat{\boldsymbol{\mu}}_m, \hat{\boldsymbol{\Sigma}}_m) > \hat{\pi}_n \mathcal{N}(\boldsymbol{x}_{r,j}; \hat{\boldsymbol{\mu}}_n, \hat{\boldsymbol{\Sigma}}_n); \forall j \neq i \qquad i, j = 1...R$$

Otherwise, $y_m^r = 0$. This means that the click $\boldsymbol{x}_{r,i}$ is associated to the m^{th} Gaussian if the maximum a posteriori probability of that click is the highest among the set of $\mathcal{X}_r$, which is the set of clicks of the r-gamer and also it is the highest with respect to the m^{th} Gaussian component, $\forall m \in \{1, ...\hat{M}\}$. As it has been explained in the Detection Stage chapter, $\boldsymbol{y}_m$ can be seen as a subset of R binary labels given by the gamers to the m^{th} parasite, belonging to the set $\mathcal{Y} = \{\boldsymbol{y}_1, ..., \boldsymbol{y}_{\hat{M}}\}$. Then these are the vectors that are used in the EM detection algorithm.

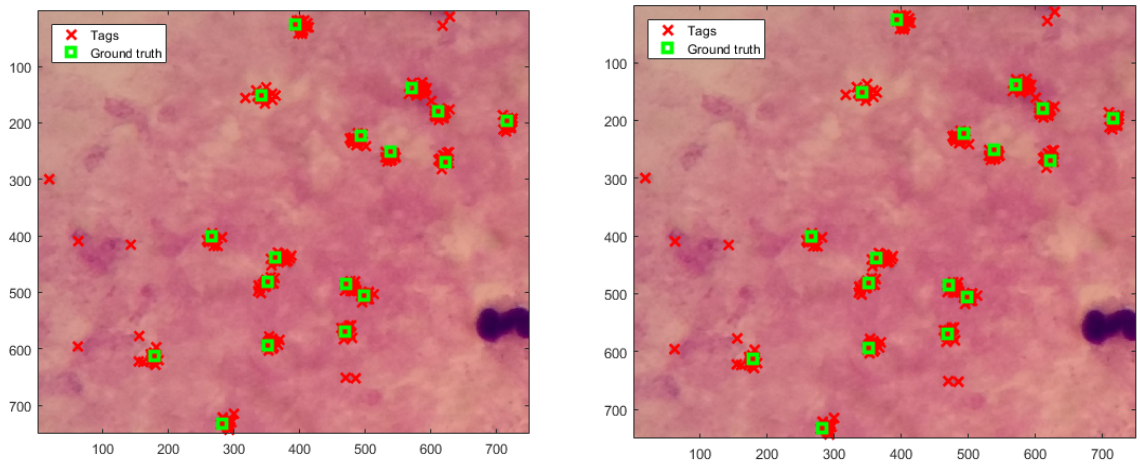
In order to test the system, in this work there have been used three real images of blood containing Malaria parasites. The first two images are the ones presented in the Clustering chapter (I1 is figure 2.5 and I2 figure 2.8) and then in here it has been added a new one (I3) in order to have more results with real data. To analyze the results, TP, TN, FP and FN have been counted over a number of realizations $N = 1000$ with a number of gamers $R = 21$ and $R = 51$. In each of the trials, after applying the OEM algorithm at the clustering stage, the Classical/Batch EM has been checked and compared with respect to the well known majority voting (MV) rule in terms of sensitivity and precision, defined in previous chapters. There is a strong relation between these parameters in both stages, i.e. at the detection stage the precision is improved at the expense of worsen the sensitivity obtained at the clustering stage.

Before showing the results, it is convenient to see the new image. It has 17 parasites and the following figures show the image, its ground truth and the tags delivered by 21 and 51 annotators.



(a) (b)

Figure 4.1: In (a) the real blood sample containing the parasites and in (b) the same but with the ground truth.



(a) (b)

Figure 4.2: In (a) tags and ground truth of a realization with 21 gamers is shown and in (b) the same but with 51 annotators.

Once the three images have been shown, the results obtained are summarized in Table 4.1.

Remembering the clustering results of I1, the sensitivity for both scenarios in terms of number of gamers, was near 1 meaning that the clustering performed well. On the contrary, the precision did not achieve 0.5, and that is why OEM found more potential parasites than the real ones. During the rest of the work, sensitivity will play a more important roll because high S means that both steps perform well, and the precision is more conditioned to the clustering because of the number of FP that OEM finds. Low S rates after the detection can be produced because there are not enough clicks in a cluster or because the algorithm does not rely on the gamers, due to that they tag lots of points in the image and in the end it provides a low estimate of the reliability parameters.

Table 4.1: Detection Sensitivity & Precision. Images I1 , I2 and I3

Detection Stage	S 21 gamers	P 21 gamers	S 35 gamers	P 35 gamers	S 51 gamers	P 51 gamers
MV I1	0.6483	0.8195	0.6350	0.8460	0.5724	0.8645
EM I1	0.7208	0.7772	0.7742	0.7441	0.8253	0.8017
MV I2	0.9733	0.9829	0.9386	0.9582	0.9421	0.9679
EM I2	0.9386	0.9992	0.9043	0.9787	0.9429	0.9729
MV I3	0.9512	0.9878	0.8941	0.9163	0.7818	0.8491
EM I3	0.8852	0.9974	0.7941	0.9569	0.7547	0.8676

For I2, the clustering performed well, obtaining high values of S and P. It can be seen in the previous table that the complete system provides similar results when using MV or EM in the detection step. EM performs better when it has a considerable number of gamers, whereas MV for this image when having 21 gamers has a greater S rate. Despite that, the values are similar and high, which means that the whole system has a good performance in images like I2. So, the main difference between the two images is that when the clustering performs very well, i.e. for I2, EM and MV provide similar results. However, if the first step is not very good, i.e. I1, EM outperforms MV.

Finally, I3 is similar to I2 in terms of color and other image features, meaning that the results do not differ too much from the ones obtained for I2. High S and P values are achieved for both configurations, what implies that both EM and MV could be used for this kind of images.

Chapter 5

On-line

5.1 Recursive EM

The algorithm that has been developed in order to be implemented in an on-line way is the Recursive EM. The other that was presented in previous sections, the Incremental Newton, has been discarded due to its performance, because apart from its slow convergence, the Recursive EM achieves better results in terms of MSE and other metrics such as Sensitivity or Precision. As in the previous section, it is going to be compared with respect to the Batch EM, that uses all the data available at each iteration and also converges to a maximum.

This algorithm will make a decision $\{0, 1\}$ on each of the M clusters as the previous ones made. In each time instant, t, the system has a binary matrix, $\boldsymbol{Y}(t)$, with the decisions that a set of gamers has delivered to each one of the clusters. When a new annotator arrives at the system $(t+1)$, his/her decisions are added at this matrix, so its size is increasing over time. This algorithm will start with a minimum number of gamers R_{min} and will end having R_{max} gamers, so for $t = 1$ the size of the matrix will be $M \times R_{min}$ and for $t = 1+(R_{max}-R_{min})$ its size will be $M \times R_{max}$. Each gamer delivers a decision vector $\boldsymbol{y}^r(t) \in \mathbb{Z}_2^{M\times 1}$ and the model of the vectors $\boldsymbol{y}_m(t) \in \mathbb{Z}_2^{R\times 1}$,$R \in [R_{min}, R_{max}]$ is the same as for the previous algorithms, a mixture of two R-dimensional Bernoulli distributions. In the end, the algorithm will provide an estimate of the parameter vector: $\boldsymbol{\theta} = [\mu, \boldsymbol{\rho}_1, \boldsymbol{\rho}_0]^T$, whose parameters mean the same as in previous sections but with the difference that over time the size of $\boldsymbol{\theta}$ is increasing. For example, for $t = 1$, $\boldsymbol{\theta} = [\mu, \rho_1^1, ...\rho_1^{R_{min}}, \rho_0^1, ..., \rho_0^{R_{min}}]$ but for $t = 1 + (R_{max} - R_{min})$ $\boldsymbol{\theta} = [\mu, \rho_1^1, ...\rho_1^{R_{min}}, ..., \rho_1^{R_{max}}, \rho_0^1, ...\rho_0^{R_{min}}, ..., \rho_0^{R_{max}}]$. In the end, the likelihood function of the observed data, $\mathcal{Y}(t)$ considering M clusters, can be written as:

$$f(\mathcal{Y}(t); \boldsymbol{\theta}) = \prod_{m=1}^{M} \Big(\mu \prod_{l=1}^{t} f(\boldsymbol{y}_m(l)|\omega = 1; \boldsymbol{\rho}_1) + (1-\mu) \prod_{l=1}^{t} f(\boldsymbol{y}_m(l)|\omega = 0; \boldsymbol{\rho}_0)\Big) =$$

$$\prod_{m=1}^{M} \Big(\mu \prod_{l=1}^{t} \prod_{r=1}^{R} (\rho_1^r)^{y_m^r} (1-\rho_1^r)^{(1-y_m^r)} + (1-\mu) \prod_{l=1}^{t} \prod_{r=1}^{R} (\rho_0^r)^{(1-y_m^r)} (1-\rho_0^r)^{y_m^r}\Big) \tag{5.1}$$

When a new gamer arrives at the system and deliver its decision vector, so the temporal variable increases in one unit, the new likelihood function yields to the following expression:

$$f(\mathcal{Y}(t+1); \boldsymbol{\theta}) = \prod_{m=1}^{M} \Big(\mu \prod_{l=1}^{t+1} f(\boldsymbol{y}_m(l)|\omega = 1; \boldsymbol{\rho}_1) + (1-\mu) \prod_{l=1}^{t+1} f(\boldsymbol{y}_m(l)|\omega = 0; \boldsymbol{\rho}_0)\Big) =$$

$$\prod_{m=1}^{M} \Big(\mu \prod_{l=1}^{t+1} \prod_{r=1}^{R+1} (\rho_1^r)^{y_m^r(l)} (1-\rho_1^r)^{(1-y_m^r(l))} + (1-\mu) \prod_{l=1}^{t+1} \prod_{r=1}^{R+1} (\rho_0^r)^{(1-y_m^r(l))} (1-\rho_0^r)^{y_m^r(l)}\Big) \tag{5.2}$$

As well as for the other EM algorithms, in order to maximize 5.2 it is needed to add latent variables, $v_m = \{0, 1\}$, with the same meaning as in the previous chapters. The complete likelihood function has the same expression as 3.10, and when a new gamer arrives at the system the new likelihood function ends being:

$$f(\mathcal{Y}(t+1), \mathcal{V}; \boldsymbol{\theta}) = f(\mathcal{Y}(t+1)|\mathcal{V}; \boldsymbol{\theta}) f(\mathcal{V}; \boldsymbol{\theta}) =$$
$$\prod_{m=1}^{M} \left(\mu \prod_{l=1}^{t+1} f(\boldsymbol{y}_m(l)|\omega = 1; \boldsymbol{\rho}_1) \right)^{v_m} \left((1-\mu) \prod_{l=1}^{t+1} f(\boldsymbol{y}_m(l)|\omega = 0; \boldsymbol{\rho}_0) \right)^{(1-v_m)} =$$
$$\prod_{m=1}^{M} \left(\mu \prod_{l=1}^{t+1} \prod_{r=1}^{R+1} (\rho_1^r)^{y_m^r(l)} (1-\rho_1^r)^{(1-y_m^r(l))} \right)^{v_m} \left((1-\mu) \prod_{l=1}^{t+1} \prod_{r=1}^{R+1} (\rho_0^r)^{(1-y_m^r(l))} (1-\rho_0^r)^{y_m^r(l)} \right)^{(1-v_m)}$$

The final algorithm is summed in Algorithm 5:

Algorithm 5 On-line Recursive EM

1: Given a number of initial gamers R_{min} initialize $\hat{\mu}, \hat{\boldsymbol{\rho}}_1$ and $\hat{\boldsymbol{\rho}}_0$
2: **for** $t = 1$ to $t = 1 + (R_{max}\text{-}R_{min})$ **Do**
3: **if** $t > 1$ **then**
4: Initialize $\rho_1^{R_{min}+t-1} = f(\boldsymbol{v}(t-1))$ and $\rho_0^{R_{min}+t-1} = f(\boldsymbol{v}(t-1))$
5: **End if**
6: *E-Step*: given an estimate $\hat{\boldsymbol{\theta}}(t)$, compute the conditional expectation:

$$Q(\boldsymbol{\theta}; \hat{\boldsymbol{\theta}}(t)) = \mathbb{E}_{\mathcal{V}}\{\log f(\mathcal{Y}(t), \mathcal{V}; \boldsymbol{\theta}) | \hat{\boldsymbol{\theta}}(t), \mathcal{Y}(t)\} = \sum_{\mathcal{V}} f(\mathcal{V}|\mathcal{Y}(t), \boldsymbol{\theta}) \log f(\mathcal{Y}(t), \mathcal{V}; \boldsymbol{\theta})$$

7: *M-Step*: obtain the estimate for the next iteration as:

$$\hat{\boldsymbol{\theta}}(t+1) = \arg\max_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}; \hat{\boldsymbol{\theta}}(t))$$

8: **End for**

It must be said that previous estimates of $\hat{v}_m(t)$ are used when a new annotator delivers to the system its decision vector $\boldsymbol{y}^r(t) \in \mathbb{Z}_2^{M \times 1}$ in order to compute its reliability parameters, so implicitly there is a reuse of the parameter vector calculated at t in order to compute the new parameter vector at $t+1$ and the values of the latent variables.

As the conditional expectation is the same as for the Recursive EM, the values for $\hat{v}_m(t)$ and the parameter estimates follow the same expressions, which are:

$$\hat{v}_m(t) = \frac{\hat{\mu}(t) \prod_{l=1}^{t} f(\boldsymbol{y}_m(l)|\omega = 1; \hat{\boldsymbol{\rho}}_1(t))}{\hat{\mu}(t) \prod_{l=1}^{t} f(\boldsymbol{y}_m(l)|\omega = 1; \hat{\boldsymbol{\rho}}_1(t)) + (1-\hat{\mu}(t)) \prod_{l=1}^{t} f(\boldsymbol{y}_m(l)|\omega = 0; \hat{\boldsymbol{\rho}}_0(t))} \tag{5.3}$$

$$\frac{\partial Q(\boldsymbol{\theta}; \hat{\boldsymbol{\theta}}(t))}{\partial \mu} = 0 \Longrightarrow \hat{\mu}(t+1) = \frac{1}{M} \sum_{m=1}^{M} \hat{v}_m(t) \tag{5.4}$$

$$\frac{\partial Q(\boldsymbol{\theta}; \hat{\boldsymbol{\theta}}(t))}{\partial \rho_1^r} = 0 \Longrightarrow \hat{\rho}_1^r(t+1) = \frac{\sum_{m=1}^{M} \hat{v}_m(t) \sum_{l=1}^{t} y_m^r(l)}{t \sum_{m=1}^{M} \hat{v}_m(t)}; \tag{5.5}$$

$$\frac{\partial Q(\boldsymbol{\theta}; \hat{\boldsymbol{\theta}}(t))}{\partial \rho_0^r} = 0 \Longrightarrow \hat{\rho}_0^r(t+1) = \frac{\sum_{m=1}^{M} (1-\hat{v}_m(t)) \sum_{l=1}^{t} (1-y_m^r(l))}{t \sum_{m=1}^{M} (1-\hat{v}_m(t))} \tag{5.6}$$

5.2 Results

In order to evaluate the performance of the algorithm, some tests with synthetic and real data have been done. First of all, the ones with synthetic data are presented and explained. As well as for the Detection Stage section, the more relevant aspects of the algorithms are the MSE and both Sensitivity and Precision, that have been computed in the same way as in that section. Different scenarios have been proposed and in all of them the number of realizations has been $N = 1000$ and the Recursive EM has been compared with respect to the Batch EM. For all the synthetic tests, the initial number of gamers is $R_{min} = 10$ and the final is $R_{max} = 50$.

- First scenario: in here it is supposed a number of clusters $M = 15$, $\mathbb{E}\{\boldsymbol{\rho}_1\} = 0.9$, $\mathbb{E}\{\boldsymbol{\rho}_0\} = 0.7$ and finally $\mu = 0.5$ representing equiprobability in each one of the clusters, which is an assumption that has been done for all the simulations.

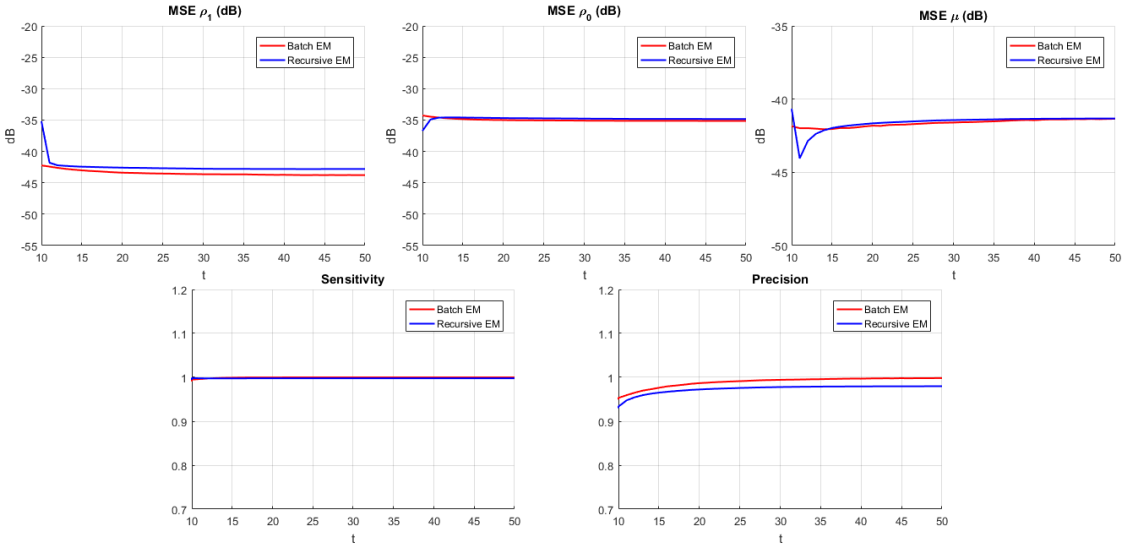


Figure 5.1: MSE of the parameter estimates and both Sensitivity and Precision rates as a function of time for $M = 15$. As it can be seen both algorithms converge fast but always the Batch EM has better results.

In here, gamers provide good decisions conditioned to one and also conditioned to zero, so that is why both S and P achieve good results, but always Batch EM is over the Recursive. Focusing on the MSE's obtained, they converge to a value once the system has approximately 25 gamers. There is little difference between both algorithms and the computational cost for Batch is higher than for EM, and this is caused because the Recursive is capable to compensate the lower computational cost by using parameters calculated in previous time instants whereas Batch starts from the scratch every time. There is a problem when using Recursive EM, because it is subject to previous parameter estimates, and if the error is high, it is difficult to reduce it over time. This phenomenon will be appreciated in the third scenario, where the gamers are not as good as in here.

- Second scenario: for the second simulation the number of clusters is $M = 10$, $\mathbb{E}\{\boldsymbol{\rho}_1\} = 0.9$ and $\mathbb{E}\{\boldsymbol{\rho}_0\} = 0.7$.

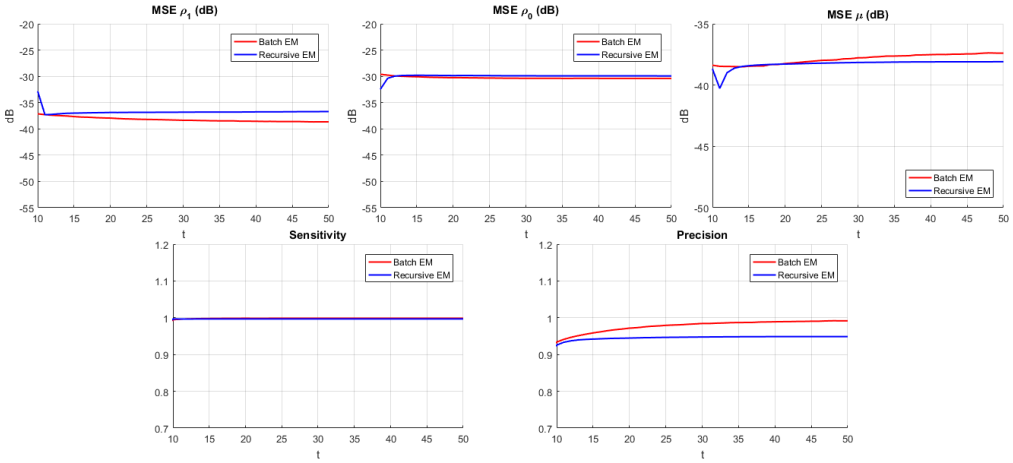


Figure 5.2: MSE of the parameter estimates and both Sensitivity and Precision rates as a function of time for $M = 10$.

The difference with respect to the first scenario is that the number of clusters has been reduced and this causes a separation between both algorithms. The system has less data, and so the Recursive has more difficulties to estimate the parameters well, while for Batch this is not a problem by its initialization from the scratch each time instant.

- Third scenario: for the last simulation the number of clusters is $M = 15$, $\mathbb{E}\{\boldsymbol{\rho}_1\} = 0.8$ and $\mathbb{E}\{\boldsymbol{\rho}_0\} = 0.6$.

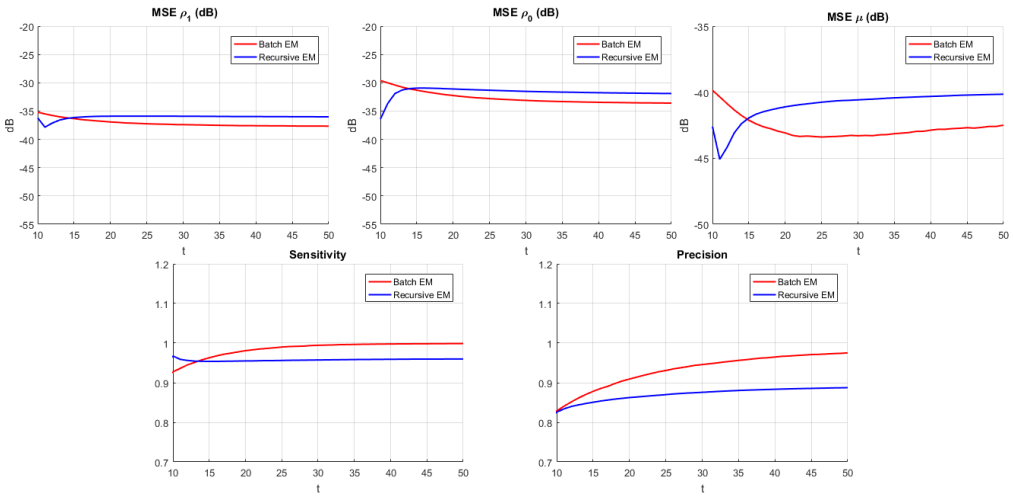


Figure 5.3: MSE of the parameter estimates and both Sensitivity and Precision rates as a function of time for $M = 15$ but decreasing the reliability parameters.

For the last simulation, the parameters of the gamers are worse than in the first scenario. This is the reason why in here, the performance of the algorithms is not as good as in the first simulation, but also the difference is higher in terms of MSE and the rates. Recursive EM is always much worse, and this is an important conclusion, that when the reliability of the gamers decreases, Batch EM outperforms Recursive EM with a considerable difference.

Finally, the on-line implementation of the Detection stage has been tested with real data. In here, the three images presented in previous chapters have been used. First of all, the clustering has been done and after that these on-line algorithms have been executed. Despite that for the synthetic simulations the algorithms have been run up to $R_{max} = 50$, in here $R_{max} = 25$ because of a matter of time in order to make a number of realizations of $N = 500$. Moreover, as the system does not know the gamers' parameters it makes no sense to calculate the MSE, so in this part only the sensitivity and precision rates are presented. The results for the first image are:

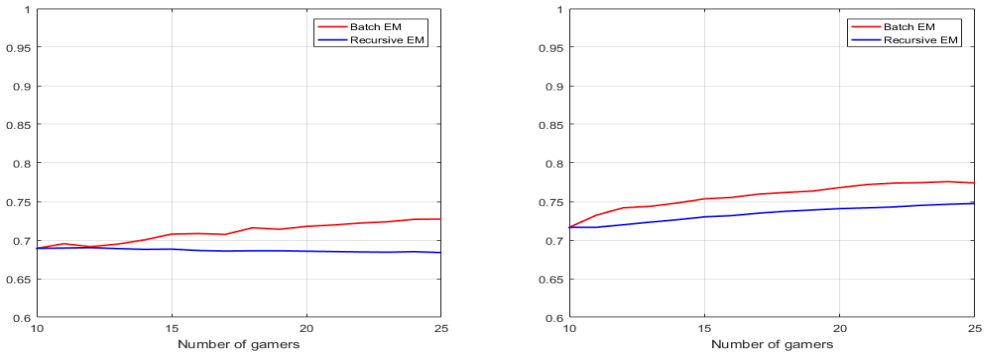


Figure 5.4: Sensitivity (left) and precision (right) for I1 from $R_{min} = 10$ to $R_{max} = 25$.

As seen in the previous image, Batch EM has a better performance for both rates despite the fact that the difference is no greater than 0.05. Table 4.1 shows the results of the two-stage algorithm, and comparing it with the on-line implementation of the Recursive EM, the second one achieves similar results but without the necessity of having to start from the scratch, which is an advantage due to that it can save time and computational cost. This little difference is caused because in here the number of gamers for the clustering is $R = 25$ whereas in the table is $R = 21$. By repeating the process but with the second image showed in previous chapters, the results are:

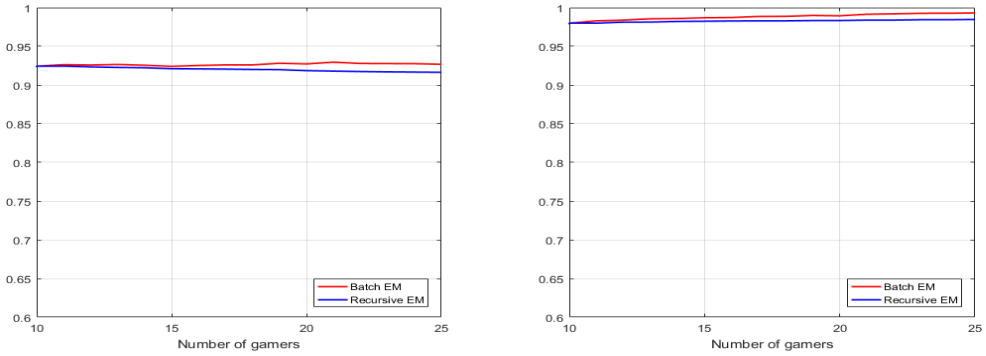


Figure 5.5: Sensitivity (left) and precision (right) for I2 from $R_{min} = 10$ to $R_{max} = 25$.

As Figure 5.5 shows, for this image the difference between both algorithms is approximately 0.01, which proves that Recursive EM can provide very similar results as Batch EM, what implies that in an on-line system in each iteration it is not needed to calculate again all the variables, so there will be an implicit reuse of the estimates made in previous time instants. Apart from that, the achieved rates are high, with a $S > 0.9$ and a $P > 0.95$. This is caused because this image has some differences with respect to I1, which makes that in here is not as difficult as for I1 to detect parasites. These differences are mainly the color and that the number of artifacts that make doubt gamers is lower.

Finally, the results obtained for I3 are showed in the following figure:

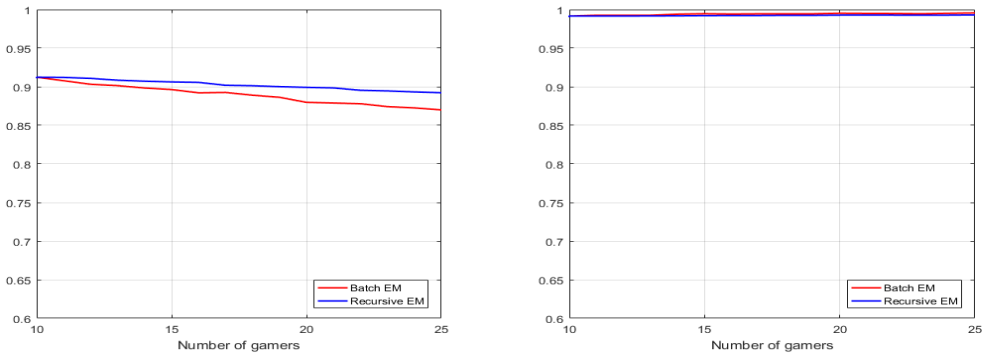


Figure 5.6: Sensitivity (left) and precision (right) for I3 from $R_{min} = 10$ to $R_{max} = 25$.

For the last image, the performance for both rates is similar to I2. The main difference is that Recursive EM outperforms Batch EM in the sensitivity, but in general terms their results are similar because the images have little differences in terms of color, shape and other characteristics.

The main conclusion obtained from this chapter, is not only that the algorithms achieve good rates, but also that Recursive EM tends to provide similar results as Batch EM, and in some cases, better ones, which proves that this algorithm can be implemented in an on-line system with lower computational cost and in a short period of time in comparison with Batch EM.

Chapter 6

Budget

In this chapter the total cost of the studies performed in this work as well as the implementation of the software, taking into account the licenses, the servers and the wages is estimated.

As said in the introduction, the work has been performed using Matlab. At least, it is needed one license in order to perform all the simulations. Each Matlab license has a validity of one year, so it is needed to renovate it each year. Apart from the software, there is another important aspect to take into account, which are the servers. All the simulations, due to its high computational cost because of the elevated number of realizations, have been done in a typical server. The following table summarizes its main features and prize of the server used in this work:

Table 6.1: Server features

Architecture	Operating System	Number of CPUs	Speed of CPUs	RAM	Approximated price
64 bits	Linux	10	10	30	60 €

Finally, the last part to be taken into account for the budget are the wages. The study and implementation of the algorithms has been developed by a junior engineer during half a year and in part-time. The work started February 6th and ended on June 25th, which involves 20 weeks at a part-time dedication (4 hours per day approximately). The typical wage for a junior engineer is about 17 €/h, so the total price for the worked hours is:

$$17\ €/h \cdot 4h/day \cdot 5\ days/week \cdot 20\ weeks = 6800\ €$$

Apart from the junior engineer's wage, the coordinators' one must be taken into account too. This work has been directed by two co-directors and during the 20 weeks there has been a dedication of 4 hours per week. The typical wage for a licensed teacher at the UPC is about 42 €/h, so the total price for the worked hour is:

$$42\ €/h \cdot 4h/week \cdot 20\ weeks \cdot 2\ co\text{-}directors = 6720\ €$$

By summing each one of the concepts of the work, it is obtained that the total cost of the project is about 13580 €.

Chapter 7

Conclusions

This final chapter concludes the work by exposing the main conclusions obtained during this period. Chapter 2 explained some clustering methods for crowdsourced data based on EM algorithms. In all the cases the algorithm that achieved better results was OEM, due to that this algorithm apart from modeling the mixture as the addition of Gaussians, it takes into account possible outliers, what makes the algorithm more robust.

Chapter 3 exposed different EM-based algorithms for the detection stage, which were Batch, Recursive and Incremental Newton. Of course Batch was included to compare the performance of the other two presented algorithms. This chapter helped us to discard Incremental Newton for the detection stage because it has a slower speed convergence and provides worse results than Recursive EM. Moreover, depending on the scenario, we could see that Recursive EM has a similar performance than Batch. For example, when the number of clusters is greater than the number of gamers, Recursive performs quite well because the system has a lot of data and the number of parameters to be estimated is not too large. So this chapter also helped to decide which algorithm was going to be tested for the on-line system, which is the Recursive.

Then, in Chapter 4, the complete off-line system was presented. This chapter provides the results in terms of sensitivity and precision for three real images and so the main conclusion that can be obtained from this chapter is that for images similar to I2 or I3, there is little difference between using MV or EM for the detection stage, but when the image has similar conditions to I1, EM outperforms MV. Apart from that, we could see the importance of having a good clustering algorithm because if it does not find all the real parasites, degrades the rates of the detection stage.

Finally, Chapter 5 implements the on-line in the detection stage. For synthetic data, we could see that there are scenarios in which Recursive EM achieves similar results as Batch EM, for example as shown in figure 5.1 where the system has good gamers. On the contrary, if the gamers do not provide reliable decisions, the difference between Recursive and Batch is higher as seen in figure 5.3. When using real data, as expected, Batch EM outperformed Recursive EM in the majority of cases, but the difference is not large, what makes possible to implement this algorithm in a real on-line system.

Some future lines could be to implement the clustering stage in an on-line way too. Moreover, other decision algorithms can be tested but Recursive EM performs quite well and has a low computational cost. Apart from that, more tests with real data can be done in order to have a complete on-line system that provides good results for different kind of images in terms of color, shape of the parasites and other image aspects.

Bibliography

[1] Bishop, C. M., *Pattern recognition in Machine Learning* Third ed. Springer Science, 2007

[2] Richard O. Duda, Peter E. Hart, David G. Stork, *Pattern classification* Second ed. Wiley-Interscience, 2006

[3] Raykar, V. C. and Yu, S. and Zhao, L. H. and Valadez, G. H. and Florin, C. and Bogoni, L. and Moy, L., *Learning from crowds* Journal of Machine Learning Research, Vol.11, p.69 (2015)

[4] Cabrera-Bean, M. and Diaz-Vilor, C. and Vidal, J. , *Impact of noisy annotators' reliability in a crowdsourcing system performance* European Signal Processing Conference (EUSIPCO), (2016)

[5] Cabrera Bean, M. , Pagès Zamora, A. , Diaz Vilor, C. , Postigo Camps, M. , Cuadrado Sanchez, D. , Luengo Oroz, M. , *Counting Malaria Parasites with a two-stage EM based algorithm using crowsourced data* Annual International Conference of the IEEE Engineering in Medicine and Biology Society , (2017)

[6] Pagès-Zamora, A, Giannakis, G, López-Valcarce, R, Gimenez-Febrer, P , *Robust clustering of data collected via crowdsourcing* IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), (2017)

[7] Gimenez-Febrer, P, Pagès-Zamora, A, López-Valcarce, R , *Online EM-based distributed estimation in sensor networks with faulty nodes* European Signal Processing Conference (EUSIPCO), (2016)

[8] Cappe, O. and Moulines, E., *Online EM Algorithm for Latent Data Models* Journal of Machine Learning Research, Vol.11, p.69 (2015)

[9] Figueiredo, Mario A. T. and Jain, Anil K., *Unsupervised learning of finite mixture models* IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol.24, p.381–396 (2012)

[10] M. A. Luengo-Oroz, A. Arranz, and J. Frean., *Crowdsourcing Malaria Parasite Quantification: An Online Game for Analyzing Images of Infected Thick Blood Smears* In: J Med Internet Res 14.6 (2012), e167

[11] `www.malariaspot.org/es/`

[12] S. Mavandadi, S. Dimitrov, S. Feng, F. Yu, U. Sikora, O. Yaglidere, S. Padmanabhan, K. Nielsen, A. Oz can., *Distributed Medical Image Analysis and Diagnosis through Crow d-Sourced Games: A Malaria Case Study* 2012 PLoS ONE 7(5): e37245. doi:10.1371/journal.pone.0037245

[13] S. Mavandadi, S. Feng, F. Yu, S. Dimitrov, R. Yu, A. Ozcan., *BioGames: A Platform for Crowd-Sourced Biomedical Image Analysis and Telediagnosis*

Games Health Journal 2012 October; 1(5): 373–376. doi: 10.1089/g4h.2012.0054, PMCID: PMC3665415

[14] `http://biogames.ee.ucla.edu/`

Appendices

Appendix A

Work Plan

A.1 Work Packages, Tasks and Milestones

The different work packages are:

1. WP1: Project Plan: organization of the work at the beginning of the project. Its main objective is to plan the start and end dates for the different tasks. The starting date was 18/02/2017 and the end date 06/03/2017.

2. WP2: Project Goals: define the main goals and objectives of the project. The planned start date was 20/02/2017 and the en date 01/03/2017.

3. WP3: Specifications: define the speficications that the project must satisfy, such as good results in terms of error probability, fast performance and so on. The start date was the same as for the WP2.

4. WP4: Theoretical development: theoretical developments of the formulas and algorithms that will be used during the project, such as EM, Maximum Likelihood estimators and so on. The starting date was 05/02/2017 and the end date was 15/04/2017.

5. WP5: Simulation and Testing: simulate the performance of the system. At the first stage, the clustering, with algorithms such as EM,K-Means and at the second one, the decision stage, with the EM-based algorithms. Its start and end dates are: 31/03/2017 and 12/06/2017.

6. WP6: Write up of the project: write the work and the results obtained during the project. This work package started at the very beginning, on the 01/03/2017 and ended on June, concretely at the 25/06/2017.

A.2 Gantt Diagram

This section contains the Gantt diagram of the project.

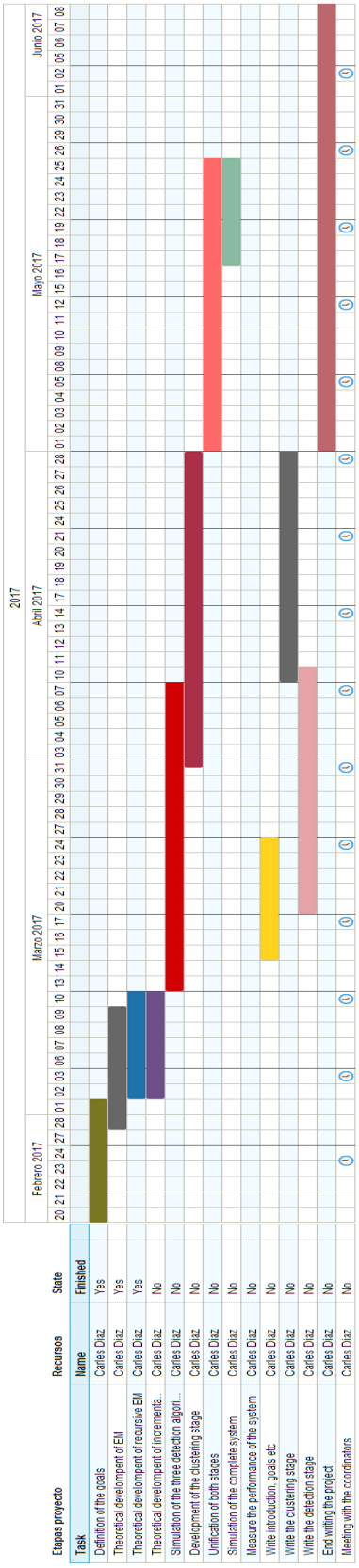


Figure A.1: Gantt diagram of the project

Appendix B

Incremental Newton

This Appendix is dedicated to the developments needed to obtain the expressions of the parameter estimates for the Incremental Newton algorithm. First of all it is going to be calculated the Fisher Information matrix, $\boldsymbol{I}(\hat{\boldsymbol{\theta}})$, whose expression is the following one:

$$\boldsymbol{I}(\boldsymbol{\theta}) = -\mathbb{E}_{\mathcal{V},\mathcal{Y}}\begin{bmatrix} \frac{\partial^2 log f(\mathcal{Y}(t),\mathcal{V};\boldsymbol{\theta})}{\partial \mu^2} & \frac{\partial^2 log f(\mathcal{Y}(t),\mathcal{V};\boldsymbol{\theta})}{\partial \mu \partial \boldsymbol{\rho}_1^T} & \frac{\partial^2 log f(\mathcal{Y}(t),\mathcal{V};\boldsymbol{\theta})}{\partial \mu \partial \boldsymbol{\rho}_0^T} \\ \frac{\partial^2 log f(\mathcal{Y}(t),\mathcal{V};\boldsymbol{\theta})}{\partial \boldsymbol{\rho}_1 \partial \mu} & \frac{\partial^2 log f(\mathcal{Y}(t),\mathcal{V};\boldsymbol{\theta})}{\partial \boldsymbol{\rho}_1 \partial \boldsymbol{\rho}_1^T} & \frac{\partial^2 log f(\mathcal{Y}(t),\mathcal{V};\boldsymbol{\theta})}{\partial \boldsymbol{\rho}_1 \partial \boldsymbol{\rho}_0^T} \\ \frac{\partial^2 log f(\mathcal{Y}(t),\mathcal{V};\boldsymbol{\theta})}{\partial \boldsymbol{\rho}_0 \partial \mu} & \frac{\partial^2 log f(\mathcal{Y}(t),\mathcal{V};\boldsymbol{\theta})}{\partial \boldsymbol{\rho}_0 \partial \boldsymbol{\rho}_1^T} & \frac{\partial^2 log f(\mathcal{Y}(t),\mathcal{V};\boldsymbol{\theta})}{\partial \boldsymbol{\rho}_0 \partial \boldsymbol{\rho}_0^T} \end{bmatrix}$$

If the log-likelihood function has the following formula:

$$\log f(\mathcal{Y}(t),\mathcal{V};\boldsymbol{\theta}) = \sum_{m=1}^{M} v_m(t) \log\left(\mu \prod_{l=1}^{t}\prod_{r=1}^{R} (\rho_1^r)^{y_m^r(l)}(1-\rho_1^r)^{(1-y_m^r(l))}\right) +$$
$$(1-v_m(t)) \log\left((1-\mu)\prod_{l=1}^{t}\prod_{r=1}^{R} (\rho_0^r)^{(1-y_m^r(l))}(1-\rho_0^r)^{y_m^r(l)}\right) =$$

$$\sum_{m=1}^{M} v_m(t)\left(\log\mu + \sum_{l=1}^{t}\sum_{r=1}^{R} y_m^r(l)\log\rho_1^r + (1-y_m^r(l))\log(1-\rho_1^r)\right) +$$

$$+\sum_{m=1}^{M}(1-v_m(t))\left(\log(1-\mu) + \sum_{l=1}^{t}\sum_{r=1}^{R}(1-y_m^r(l))\log(\rho_0^r) + y_m(l)^r\log(1-\rho_0^r)\right) \tag{B.1}$$

we do not need to calculate the cross derivatives because the first partial derivatives do not depend on other parameters. So it is just needed to make the second partial derivatives and, after that, do the expected value with respect to $\mathcal{V}$ and $\mathcal{Y}$.

The expressions for the parameter μ are:

$$\frac{\partial \log f(\mathcal{Y}(t),\mathcal{V};\boldsymbol{\theta})}{\partial \mu} = \sum_{m=1}^{M}\left(\frac{v_m(t)tR}{\mu} + \frac{-(1-v_m(t))tR}{1-\mu}\right)$$

$$\frac{\partial^2 \log f(\mathcal{Y}(t),\mathcal{V};\boldsymbol{\theta})}{\partial \mu^2} = \sum_{m=1}^{M}\left(\frac{-v_m(t)tR}{\mu^2} + \frac{-(1-v_m(t))tR}{(1-\mu)^2}\right)$$

$$-\mathbb{E}_{\mathcal{V},\mathcal{Y}}\{\frac{\partial^2 \log f(\mathcal{Y}(t),\mathcal{V};\boldsymbol{\theta})}{\partial \mu^2}\} = \sum_{m=1}^{M}\left(\frac{tR}{\mu^2}\mu + \frac{tR}{(1-\mu)^2}(1-\mu)\right) = \frac{tRM}{\mu(1-\mu)}$$

Doing the same as for μ, the expressions that allow to calculate $\hat{\rho}_1^r$ and $\hat{\rho}_0^r$ are:

$$\frac{\partial \log f(\mathcal{Y}(t), \mathcal{V}; \boldsymbol{\theta})}{\partial \rho_1^r} = \sum_{m=1}^{M} v_m(t) \sum_{l=1}^{t} \left(\frac{y_m^r(l)}{\rho_1^r} - \frac{1 - y_m^r(l)}{1 - \rho_1^r} \right)$$

$$\frac{\partial^2 \log f(\mathcal{Y}(t), \mathcal{V}; \boldsymbol{\theta})}{\partial \rho_1^{r2}} = \sum_{m=1}^{M} v_m(t) \sum_{l=1}^{t} \left(\frac{-y_m^r(l)}{\rho_1^{r2}} - \frac{1 - y_m^r(l)}{(1 - \rho_1^r)^2} \right)$$

$$-\mathbb{E}_{\mathcal{V},\mathcal{Y}}\{\frac{\partial^2 \log f(\mathcal{Y}(t), \mathcal{V}; \boldsymbol{\theta})}{\partial \rho_1^{r2}}\} = \sum_{m=1}^{M} \mu \sum_{l=1}^{t} \left(\frac{\rho_1^r}{\rho_1^{r2}} + \frac{(1 - \rho_1^r)}{(1 - \rho_1^r)^2} \right) = \frac{t\mu M}{\rho_1^r (1 - \rho_1^r)}$$

$$\frac{\partial \log f(\mathcal{Y}(t), \mathcal{V}; \boldsymbol{\theta})}{\partial \rho_0^r} = \sum_{m=1}^{M} (1 - v_m(t)) \sum_{l=1}^{t} \left(\frac{1 - y_m^r(l)}{\rho_0^r} - \frac{y_m^r(l)}{1 - \rho_0^r} \right)$$

$$\frac{\partial^2 \log f(\mathcal{Y}(t), \mathcal{V}; \boldsymbol{\theta})}{\partial \rho_0^{r2}} = \sum_{m=1}^{M} (1 - v_m(t)) \sum_{l=1}^{t} \left(\frac{-(1 - y_m^r(l))}{\rho_1^{r2}} - \frac{y_m^r(l)}{(1 - \rho_0^r)^2} \right)$$

$$-\mathbb{E}_{\mathcal{V},\mathcal{Y}}\{\frac{\partial^2 \log f(\mathcal{Y}(t), \mathcal{V}; \boldsymbol{\theta})}{\partial \rho_0^{r2}}\} = \sum_{m=1}^{M} (1 - \mu) \sum_{l=1}^{t} \left(\frac{\rho_0^r}{\rho_0^{r2}} + \frac{(1 - \rho_0^r)}{(1 - \rho_0^r)^2} \right) = \frac{t(1 - \mu)M}{\rho_0^r (1 - \rho_0^r)}$$

By filling the matrix with these expressions, its formula is:

$$\boldsymbol{I}(\hat{\boldsymbol{\theta}}) = \begin{bmatrix} \frac{tRM}{\hat{\mu}(1-\hat{\mu})} & 0 & \dots & \dots & \dots & \dots & 0 \\ 0 & \frac{t\mu M}{\hat{\rho}_1^1(1-\hat{\rho}_1^1)} & \dots & \dots & \dots & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & \dots & \frac{t\mu M}{\hat{\rho}_1^1(1-\hat{\rho}_1^R)} & \dots & \dots & 0 \\ 0 & \dots & \dots & \dots & \frac{t(1-\mu)M}{\hat{\rho}_0^1(1-\hat{\rho}_0^1)} & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & \dots & \dots & \dots & 0 & \frac{t(1-\mu)M}{\hat{\rho}_0^R(1-\hat{\rho}_0^R)} \end{bmatrix}$$

There is another term needed to calculate the Newton update, which is $\mathbb{E}_{\mathcal{V}}\{\nabla_{\hat{\boldsymbol{\theta}}} log f(\mathcal{Y}(t), \mathcal{V}; \boldsymbol{\theta}) | \mathcal{Y}(t)\}$. By using the logarithms needed for the Fisher matrix, we can calculate this expression for each parameter:

$$\mathbb{E}_{\mathcal{V}}\{\nabla_{\hat{\mu}(t)} log f(\mathcal{Y}(t), \mathcal{V}; \boldsymbol{\theta}) | \mathcal{Y}(t)\} = \frac{tR}{\hat{\mu}(t)(1 - \hat{\mu}(t))} \sum_{m=1}^{M} \hat{v}_m(t) - \hat{\mu}(t)$$

$$\mathbb{E}_{\mathcal{V}}\{\nabla_{\hat{\rho}_1^r(t)} log f(\mathcal{Y}(t), \mathcal{V}; \boldsymbol{\theta}) | \mathcal{Y}(t)\} = \sum_{m=1}^{M} \hat{v}_m(t) \sum_{l=1}^{t} \frac{y_m^r(l) - \hat{\rho}_1^r(t)}{\hat{\rho}_1^r(t)(1 - \hat{\rho}_1^r(t))}$$

$$\mathbb{E}_{\mathcal{V}}\{\nabla_{\hat{\rho}_0^r(t)} log f(\mathcal{Y}(t), \mathcal{V}; \boldsymbol{\theta}) | \mathcal{Y}(t)\} = \sum_{m=1}^{M} (1 - \hat{v}_m(t)) \sum_{l=1}^{t} \frac{1 - y_m^r(l) - \hat{\rho}_0^r(t)}{\hat{\rho}_0^r(t)(1 - \hat{\rho}_0^r(t))}$$

The final expression of the estimates is in the corresponding section.